Vom Fachbereich für Mathematik und Informatik
der Technischen Universität Braunschweig
genehmigte Dissertation
zur Erlangung des Grades eines
Doktor-Ingenieurs (Dr.-Ing.)

Jobst Michael Löffler

Die Software-Architektur SCA3D

Adaptive 3D Visualisierung und Kooperation in offenen Informationsräumen

Datum der Promotion (Tag der mündlichen Prüfung): 16. Oktober 2002

1. Referent: Prof. Dr. Dieter W. Fellner
2. Referent: Prof. Dr. Hans-Jürgen Appelrath
eingereicht am: 20. März 2002

Abstract

This work presents the software architecture SCA3D as a flexible framework for distributed 3D visualization which allows an extended view on new concepts for management of complex and protected 3D documents. Digital data have a determining influence on the modern work flow from the first conception phase to the final manufacturing of a product. In most cases the work flow is a distributed process in which multiple users work together in an environment of networked computers. Requirements for distributed work with digital data, which are regarded as digital documents, are the following: First, well-defined interfaces between processing systems are needed to allow cooperation and data exchange. Second, methods for long-term storage in archives, for retrieval and for distribution of digital documents are required which are able to manage huge amounts of complex digital documents. Third, adaptation of multimedia documents during distribution in heterogeneous environments must be supported to control network and local processing load.

Digital libraries, understood as open information spaces, offer a suitable framework for describing solutions that fulfill the above requirements. Here the concept of an open information space is used in the context of digital libraries to emphasize two points: First, that digital library services are not restricted to closed groups of users and information servers. Second, that it is important that uniform conditions for access to digital documents are provided throughout an information space and that interfaces to other information spaces can be established.

The main focus of the work presented here is the exploration of new methods for distributed processing of 3D documents in open information spaces. Three dimensional scenes, as for example CAD models, are considered to be digital documents containing structure information, geometric content and descriptive metadata. Examples of applications are: distributed development of new products in the automotive industry using digital mock-ups, medical information systems or cooperative learning environments. On the one hand it can be observed that the number of 3D documents exchanged between applications is growing from year to year and that their complexity increases. On the other hand modern working environments are becoming more heterogeneous. The spectrum of systems involved in a work process can range from powerful workstations in a local network to mobile end devices with restricted processing power and connection bandwidth.

As a response to this challenge this dissertation presents a software architecture for distributed 3D visualization that supports document features in open distributed environments and provides functionality for management of complexity and security requirements of 3D documents. The status quo of the research on distributed software architectures for 3D visualization is extended by this work in two ways. First, it develops a new approach to supporting hybrid 2D/3D client scenes in distributed and cooperative work processes. Second, it introduces new methods for complexity and security management that allow distribution of the processing load (rendering) and individual control of the network load. In this context a new method is also presented which enables application independent access to distributed 3D documents (*document request broker*). This work describes the *level-of-information* approach that was developed to make it possible to adapt transmission rate and local processing load using a performance metric. Finally, a graded rights concept is part of the software architecture. This concept guarantees the protection of document data by controlled dissemination of copies with adapted information content.

In practice the results of this work provide new possibilities for digital product development processes which can now use a wide scale of information about the 3D scenes being worked on. This scale ranges from descriptive metadata to the original 3D content of a document. A scenario like a distributed design review in a strongly heterogeneous environment with multiple developers working on complex and protected 3D documents is now supported in a more direct way than by approaches (*full replication approach*) which have been previously applied.

Zusammenfassung

Die vorliegende Arbeit stellt mit der Software-Architektur SCA3D einen flexiblen Rahmen zur verteilten 3D Visualisierung vor, der einen erweiterten Blick auf neue Konzepte zum Management von komplexen und geschützten 3D Dokumenten erlaubt. Digitale Daten bestimmen in immer stärkerem Maß moderne Arbeitsprozesse von der ersten Konzeptionsphase bis zur Fertigung eines Produktes. Dabei handelt es sich in vielen Fällen um verteilte Prozesse, die von mehreren mitwirkenden Personen mit Hilfe vernetzter Computersysteme durchgeführt werden. Hieraus ergeben sich Anforderungen für das verteilte Arbeiten mit digitalen Daten, die nun als digitale Dokumente aufgefaßt werden: Erstens sollte es definierte Schnittstellen zwischen Verarbeitungssystemen geben, um eine Zusammenarbeit und den Datenaustausch zu ermöglichen. Zweitens werden Methoden zur Langzeitspeicherung in Archiven, zum Retrieval und zur Verteilung von digitalen Dokumenten benötigt, die in der Lage sind, große Mengen von komplexen Dokumenten handhabbar zu machen. Schließlich sollte eine Adaption der multimedialen Dokumente zur Kontrolle der Netzwerklast und lokalen Verarbeitungslast bei der Verteilung in heterogenen Umgebungen unterstützt werden.

Digitale Bibliotheken verstanden als offene Informationsräume bieten einen geeigneten Rahmen, um Lösungsansätze zur Erfüllung der genannten Anforderungen zu beschreiben. In dieser Arbeit wird der Begriff des offenen Informationsraumes für digitale Bibliotheken verwendet, da einerseits die angebotenen Dienste nicht auf vorher bekannte Benutzergruppen und Informations-Server beschränkt sind. Andererseits stehen gleichförmige Bedingungen für den Zugriff auf digitale Dokumente im gesamten Informationsraum zur Verfügung und Schnittstellen zu anderen Informationsräumen werden ermöglicht.

Der Schwerpunkt der vorliegenden Arbeit liegt auf der Untersuchung von neuen Methoden zur verteilten Verarbeitung von 3D Dokumenten in offenen Informationsräumen und deren Umsetzung in Form einer Software-Architektur. Dreidimensionale Szenen, wie z.B. CAD-Modelle, werden als digitale Dokumente mit Struktur, Inhalt und beschreibenden Metadaten aufgefaßt. Beispiele hierfür sind die verteilte Produktentwicklung im Automobilbereich mit digitalen Prototypen (Digital Mock-Up), medizinische Informationssysteme oder auch kooperative Lernumgebungen. Hierbei ist zu beobachten, dass die Anzahl der ausgetauschten 3D Dokumente und ihre Komplexität in der Praxis fortlaufend ansteigt. Andererseits werden moderne Arbeitsumgebungen immer heterogener, wobei das Spektrum beteiligter Systeme vom leistungsstarken Arbeitsplatzrechner im lokalen Netz bis hin zum mobilen Endgerät mit beschränkter Rechenleistung und Anbindungsbandbreite reichen kann.

Als Antwort auf diese Problemstellung stellt diese Arbeit eine Software-Architektur zur verteilten 3D Visualisierung vor, die Dokumenteigenschaften in offenen, verteilten Umgebungen unterstützt und ein Komplexitäts- und Sicherheitsmanagement ermöglicht. Der Stand der Forschung auf dem Gebiet verteilter Software-Architekturen zur 3D Visualisierung wird zum einen durch die Entwicklung eines neuen Ansatzes zur Unterstützung von hybriden 2D/3D Client-Szenen in verteilten, kooperativen Arbeitsprozessen erweitert. Zum anderen werden neue Methoden zur Verteilung der Verarbeitungslast (Rendering), zur Regulierung der Netzwerklast und zur Sicherung von 3D Dokumenten vorgestellt. Hierzu zählt die Entwicklung eines Verfahrens zum anwendungsunabhängigen Zugriff auf verteilte 3D Dokumente (*Document-Request-Broker*) und des *Level-of-Information*-Ansatzes, der eine Adaptivität bezüglich der Übertragungsrate und lokalen Berechnungslast mittels einer Performanzmetrik ermöglicht. Schließlich wird ein abgestuftes Rechtekonzept entworfen, das die Dokumentsicherheit durch ein kontrollierte Verteilung von Kopien mit angepaßtem Informationsgehalt gewährleistet.

Für die Praxis ergeben sich aus den Ergebnissen der Arbeit neue Möglichkeiten für Arbeitsprozesse in der digitalen Produktentwicklung, die eine breite Skala von Informationen über die 3D Szenen nutzen können, an denen gearbeitet wird. Diese Skala reicht von den beschreibenden Metadaten bis hin zu den originalen 3D Inhalten. Ein Szenario wie das des verteilten Design-Reviews von mehreren beteiligten Entwicklern in einer stark heterogenen Umgebung mit komplexen und zum Teil geschützten 3D Dokumenten wird damit in verbesserter Form unterstützt, als es in früheren Ansätzen (*Full-Replication-Ansatz*) der Fall war.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

"We must plan for freedom, and not only for security, if for no other reason than that only freedom can make security secure."

Karl R. Popper, "The Open Society and Its Enemies", 1943

Danksagung

An erster Stelle möchte ich Herrn Professor Dieter Fellner für die Betreuung der Arbeit und die zahlreichen Gespräche in Bonn und Braunschweig danken, die zum Gelingen der Arbeit und zur gemeinsamen Entwicklung vieler wichtiger Ideen der Arbeit geführt haben. Ebenfalls danken möchte ich Herrn Professor Martin Reiser, der als Leiter des Instituts für Medienkommunikation die Anfertigung der Arbeit ermöglicht und in vieler Hinsicht das Weiterkommen unterstützt hat. Als sehr engagierten Zweitgutachter habe ich Herrn Professor Hans-Jürgen Appelrath vom Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme OFFIS kennengelernt, dem ich hiermit meinen Dank aussprechen möchte. Der Betreuung innerhalb der Arbeitsgruppe Value-added-Solutions (VaS) am IMK durch Herrn Dr. Manfred Bogen ist zu verdanken, daß ich auf dem gesamten Weg von der ersten Idee bis zur fertigen Arbeit immer wieder das Ziel im Auge behalten habe. Danke für die guten Gespräche, Kommentare und Anregungen im Rahmen der Arbeit und die offene Arbeitsatmosphäre, Manfred. Die Zusammenarbeit und Diskussion des Arbeitsthemas mit Herrn Dr.-Ing. Joachim Köhler haben mir viele neue Ideen und die nötige Unterstützung für die letzte Phase der Arbeit gegeben. Herrn Dipl.-Phys. Peter Wunderling möchte ich als Arbeitsgruppenleiter des neuen Kompetenzzentrums NetMedia am Fraunhofer-IMK für seine Hilfe und Unterstützung danken.

Für die sehr gute Zusammenarbeit als Doktoranden in NetMedia und für das geduldige Korrekturlesen des fertigen Textes möchte ich besonders Gundula Dörries danken. Die mittäglichen Diskussionen bei exotischen Teesorten waren für meine Arbeit sehr anregend. In den verschiedenen Phasen der Arbeit haben mir viele weitere Kollegen geholfen: Andreas Karajannis war stets ein kompetenter Ansprechpartner im Bereich Web-Technologie und Datenbanken, Lothar Zier hat mir im weiten Gebiet der Netzwerke zahlreiche Tipps gegeben und Martha Larson hat bei der Formulierung englischer Papiere mit nützlichen Hinweisen geholfen. Durch den Einklang bei der gemeinsamen Nutzung eines Büros und zeitweilig ihres Macs hat Inke Kolb bei der Fertigstellung der Arbeit mitgewirkt. Allen anderen Mitarbeitern der Arbeitsgruppen VaS und NetMedia möchte ich noch einmal besonders für die Hilfe in fachlichen Fragen, die gute Zusammenarbeit und den Spaß bei vielen Freizeiten danken.

Schließlich geht mein Dank an meine Eltern und meine Schwestern dafür, daß sie meine Arbeit stets unterstützt und mir auch in allem, was darüber hinausgeht, immer geholfen haben.

Kapitel 1

Einführung

Mit dem Aufkommen global zugänglicher Informationssysteme, wie z.B. digitalen Bibliotheken im Internet, entstehen neue Arbeitsprozesse (z.B. virtuelle Produktdefinition), die eine Weiterentwicklung fortschrittlicher Techniken für Retrieval, Darstellung und Interaktion digitaler Objekte nötig machen. Die wachsende Verfügbarkeit und Komplexität von digitalen Dokumenten, die ebenfalls einen Schutz von Originaldaten erforderlich machen, müssen zudem Einfluß auf die Entwicklung moderner Anwendungsarchitekturen nehmen. Als spezielle Anforderung an Informationssysteme wird das interaktive Arbeiten mit 3D Szenen von zahlreichen Anwendungsbereichen wie dem Computer-Supported-Cooperative-Design, der Medizin oder dem Computer-based Training gefordert. Für die Untersuchungen und Entwicklungen dieser Arbeit werden die Eigenschaften von digitalen Dokumenten für das verteilte Arbeiten mit 3D Szenen genutzt, um einen flexiblen Umgang mit komplexen und geschützten 3D Dokumenten in offenen Informationsräumen zu ermöglichen.

Mit dem Ziel, eine flexible und offene Umgebung für das Retrieval, die Visualisierung und Manipulation von 3D Dokumenten zu schaffen, wird die verteilte Software-Architektur SCA3D als erweiterbares Mehrbenutzersystem entwickelt. Ein offener Informationsraum, in dem sich die Anwender während eines Arbeitsprozesses virtuell bewegen, umfaßt hier mehrere Informations-Server mit Dokumentdatenbanken, eine potentiell große Anzahl von Benutzern mit verschiedenen Endgeräten und Kommunikationsnetze zwischen diesen Komponenten. Die Hauptmerkmale eines offenen Informationsraumes sind die einheitlichen Bedingungen für das Arbeiten mit digitalen Objekten, die allen Benutzern trotz unterschiedlicher Umgebungsbedingungen bereitgestellt werden, und die Erweiterbarkeit durch Schnittstellen zu weiteren Informationsräumen. Um eine individuelle Skalierung der lokal verfügbaren Informationsmenge für jeden Benutzer zu ermöglichen, wurde ein Level-of-Information Ansatz entworfen und in die Software-Architektur integriert. Die gewählten Basistechnologien zur Entwicklung verteilter objekt-orientierter Software-Architekturen ermöglichen durch ihre Verbreitung und Interoperabilität einen weiten Einsatzbereich in der Praxis. Anwendungen zum inhalts-basierten Retrieval in 3D Web-Datenbanken können mit der entwickelten Software-Architektur SCA3D genauso realisiert werden, wie kooperative 3D Designumgebungen mit Anforderungen aus der Industriepraxis.

1.1 Problemstellung

Bei der verteilten interaktiven Visualisierung von 3D Szenen werden heute im wesentlichen zwei Ansätze verfolgt, nämlich einerseits das lokale Rendering von replizierten Daten [HSF99] und andererseits ein Remote-Rendering-Ansatz [ESE99], [HJT98] mit zentral verwalteten 3D Geometrie-

daten. Die Daten und Prozesse werden dabei zwischen den Systemkomponenten auf unterschiedliche Weise verteilt, wodurch die Belastung des verbindenden Netzes und der beteiligten Rechner zu Engpässen in der Übertragung und Verarbeitung führen kann.

Da ein direktes Feedback bei der 3D Visualisierung sehr entscheidend für die Benutzbarkeit eines Systems ist, wird in vielen Anwendungen der lokale Rendering-Ansatz eingesetzt, d.h. die 3D Dokumente werden als Kopie zum Benutzer übertragen und dort weiterverarbeitet. Den Vorteilen, die dieser Ansatz bietet, stehen aber auch gravierende Nachteile gegenüber, die sehr entscheidend für heutige und zukünftige Systeme sein können. Der wesentliche Nachteil lokaler Visualisierung in verteilten Anwendungen besteht darin, daß für geschützte Daten keine direkten Möglichkeiten bestehen, um die Originaldaten wirksam zu schützen. Da zur Visualisierung mit hoher Qualität in der Regel die Originaldaten nötig sind, muß immer eine vollständige Kopie dieser Daten über das Netz verteilt werden. Kennzeichung mit digitalen Wasserzeichen und Signaturen, wie es auch zur Sicherung von digitalen Bilddaten verwendet wird, ist beim heutigen Stand der Technik als nicht immer voll ausreichender Schutz von 3D Daten anzusehen, da eine nachträgliche Manipulation dieser Zusatzdaten relativ leicht möglich ist. Ein weiterer Nachteil lokaler Visualisierungsansätze ist, daß für sehr komplexe 3D Dokumente die Übertragung selbst zum Problem wird, da die Netzleistung und eventuell die lokal bereitgestellte Verarbeitungsleistung des Benutzerrechners nicht ausreichend ist. In diesem Fall gibt es heute keine Möglichkeit, die tatsächlich am Client benötigte Menge an Information für beliebige Repräsentationsformen von 3D Modellen (parametrische Modelle und polygonale Netze) stufenweise zu regulieren. Level-of-Detail-Ansätze funktionieren bisher nur für 3D-Netze, können also erst nach dem Tesselierungsschritt eingesetzt werden. Für parametrische Modelle und angereicherte zweidimensionalen Information (Bildobjekte, Bounding-Boxes, etc.), die eine Nutzung der vollständige Skala von Repräsentationen von 3D Objekten in Visualisierungsumgebungen ermöglichen, sind bisher keine Untersuchungen mit der Zielsetzung einer dynamischen Kontrolle des Informationsgehalts vorgenommen worden.

Eine alternative Lösung zum Schutz der Eigentumsrechte und zur minimalen Belastung des Netzes und der Client-Anwendung mit Geometriedaten ist der *Remote-Rendering*-Ansatz, bei dem die Daten am Server berechnet werden und am Client als Bildinformation visualisiert werden. Dieser Ansatz hat aber den Nachteil, dass immer eine mehr oder weniger große Verzögerung in den interaktiven Arbeitsprozess eingeführt wird. Hierfür ist die Verzögerung durch die Netzübertragung der Bilddaten verantwortlich, aber auch die server-seitige Bilderzeugung hinein und die Bildanzeige auf Client-Seite verursachen Verzögerungen.

Die beiden wichtigsten Probleme, die bei der Entwicklung einer verteilten Visualisierungsumgebung zu lösen sind, ergeben sich also aus der Komplexität der zu bearbeitenden 3D Daten und den hohen Sicherheitsanforderungen, die zum Schutz der Originaldaten gestellt werden. Lösungsansätze für diese Probleme sollten Methoden für ein effektives Komplexitäts- und Sicherheitsmanagement für 3D Dokumente in heterogenen Netzwerkumgebungen bereitstellen.

1.2 Lösungsansatz

Der in dieser Arbeit vorgestellte Ansatz kombiniert die Methoden des lokalen Renderings und des Remote-Renderings in verteilten Visualisierungsumgebungen, um die Vorteile der beiden Ansätze zu nutzen und neue Techniken zur Lösung der angesprochenen Probleme zu entwickeln. Dabei wird der Begriff des Level-of-Information für ein 3D Objekt eingeführt, der den verfügbaren Informationsgehalt für ein 3D Objekt auf der Skala zwischen exakter 3D Information und reiner Bildinformation beschreibt.

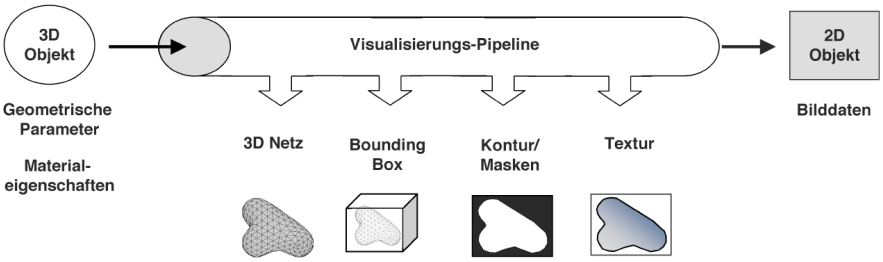

Abbildung 1.1: Repräsentationen von 3D Objekten mit verschiedenem Level-of-Information.

Durch den Level-of-Information-Ansatz ist es also möglich, daß komplexe Informationen (hier 3D Dokumente) in verteilten Umgebungen nicht von vornherein vollständig zum Benutzer übertragen werden müssen, um mit ihnen interaktiv arbeiten zu können. Vielmehr wird hier ein Ansatz verfolgt, bei dem nur die für den aktuellen Grad an Interaktion nötige Informationsmenge übertragen wird. Für 3D Dokumente bedeutet dies, daß eine schrittweise Steigerung der Informationsmenge am Client von reiner Bildinformation, über Bildinformation mit Objektkontur, über Textur plus Kontur plus Bounding-Box, bis hin zur vollständigen 3D Geometrie realisiert werden kann. In Abbildung 1.1 ist die Erzeugung verschiedener Repräsentationen von 3D Objekten mit variierendem Level-of-Information entlang einer Visualisierungs-Pipeline gezeigt.

Die skizierte Lösung der beiden genannten Probleme setzt voraus, daß die Struktur und der Inhalt der 3D Dokumente auf verschiedenen Repräsentationsebenen in dem verteilten System zugänglich gemacht werden. Zu diesem Zweck bietet sich die Verwendung Szenengraph-basierter Visualisierungsmethoden an. Allgemeiner wird im weiteren der Begriff des objekt-orientierten Dokumentmodelles verwendet, der dem Szenengraphansatz bei der 3D Visualisierung entspricht und es erlaubt, 3D Szenen als 3D Dokumente anzusehen. Der hier vorgestellte Lösungsansatz der *Skalierbaren Szenen*, der auf einem objekt-orientierten Dokumentmodell aufbaut, arbeitet mit einem Navigationsbereich auf Server-Seite (S) und einem Interaktionsbereich auf Client-Seite (C), zwischen denen sich ein IPR Filter (*Intellectual Property Rights*) zum Verteilungsmanagement von geschützten 3D Objekten befindet (siehe Abbildung 1.2). Durch Verwendung von Remote-Visualisierung im Navigationsbereich und lokaler Visualisierung im Interaktionsbereich kann auf Objekte auf verschieden Arten zugegriffen werden, je nachdem in welchem Bereich sie sich aktuell befinden. Ein Benutzer erhält also die Möglichkeit, von der reinen Navigation in einer virtuellen Szene schrittweise zu einer lokalen Interaktion zu gelangen.

1.3 Zielsetzung und Forschungsbeitrag

In der vorliegenden Arbeit wird ein Konzept und die Entwicklung einer verteilten Software-Architektur vorgestellt, die die Ansätze des Remote-Rendering und des lokalen Rendering kombiniert, um den Level-of-Information-Ansatz zu realisieren. Diese Software-Architektur erlaubt den interaktiven Zugriff auf verteilte Objekte eines 3D Dokumentes bei der Visualisierung über 2D Information (Pixel, Remote-Rendering) bzw. über die 3D Information (Geometriedaten, lokales Rendering). Zwischen diesen beiden Begrenzungspunkten werden möglichst kontinuierliche Zwischenschritte des interaktiven Zugriffs ermöglicht. Dieser Lösungsansatz wird hier mit einem Schieberegler verglichen, der zwischen den beiden extremen Repräsentationsformen eines Objektes als

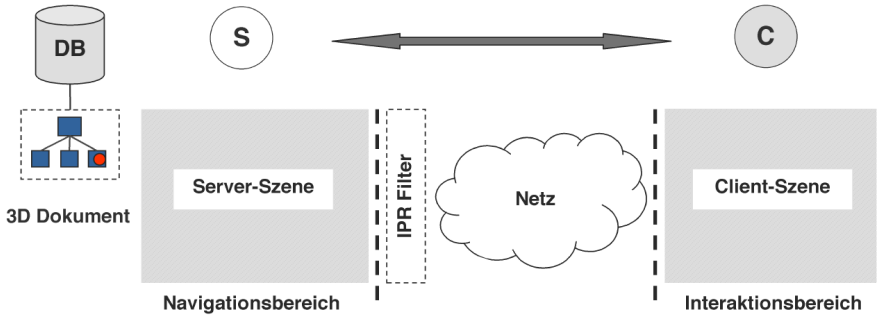

Abbildung 1.2: Systemübersicht des vorgeschlagenen Lösungsansatzes.

3D Geometriedaten bzw. als 2D Bilddaten kontinuierlich reguliert werden kann. In Abbildung 1.3 ist das Schieberegler-Prinzip innerhalb einer verteilten Visualisierungsumgebung veranschaulicht. Durch Zugriff auf die Struktur des 3D Dokumentes am Server können einzelne Objekte selektiert und schrittweise aus dem Hintergrund herausgelöst werden, womit eine schrittweise Übertragung der Objektinformation zum Client verbunden ist. Die Beschränkungen der Positionen des Schiebereglers durch die Randbedingungen der Umgebung und die dynamische Kontrolle der Position durch eine geeignete Steuerung sollen in der Arbeit untersucht werden.

Auf konzeptzioneller Ebene trägt die Arbeit mit dem Zusammenbringen der Begriffe des digitalen Dokumentes und der 3D Szene zum Forschungsbereich der multimedialen Informationssysteme bei, indem die vorhandenen Methoden zur Verarbeitung von digitalen Dokumenten und von 3D Daten kombiniert und für den zukünftigen Einsatz in digitalen Bibliotheken untersucht werden. Insbesondere wird das Potential dieser 3D Dokumente für den Umgang mit komplexer und geschützter Information in den oben beschriebenen offenen Informationsräumen untersucht. Auf technischer Ebene werden dafür die Kombinationsmöglichkeiten verschiedener Visualisierungskonzepte untersucht und eine Software-Architektur entwickelt, die eine dynamische Integration von lokalem Rendering und Remote-Rendering in einer verteilten Mehrbenutzeranwendung ermöglicht. Ein eigener Steuerungsmechanismus zur adaptiven Kontrolle des Informationsgehaltes auf Client-Seite wird entwickelt und in praktischen Tests ausgewertet. Hierfür wird eine Performanzmetrik eingeführt, die zur Steuerung des Level-of-Information-Parameters in der verteilten Visualisierungsumgebung verwendet wird. Schließlich wird eine Konzept für den Zugriff auf beliebige 3D Dokumente in verteilten Umgebungen umgesetzt, das der Idee eines Middleware-Ansatzes folgt und statische bzw. dynamisch erzeugte Schnittstellen für 3D Objekte verwendet. Damit wird eine Zugriffs- und Ortstransparenz beim Arbeiten mit 3D Dokumenten etabliert, d.h. ein Anwender stellt keinen Unterschied beim Zugriff auf lokale bzw. entfernte Objekte mehr fest und der tatsächliche Ort (Server) eines entfernten Objektes muss nicht bekannt sein. Mit dynamisch erzeugten Schnittstelleninformationen für 3D Objekte wird darüber hinaus eine Unabhängigkeit von der eingesetzten Anwendung auf Client-Seite möglich. Der Beitrag der Arbeit zum aktuellen Stand der Forschung liegt daher neben den beschriebenen Punkten der technischen Ebene besonders auch in der Entwicklungsarbeit eines zukuntsorientierten Gesamtsystems für den flexiblen Umgang mit digitalen Dokumenten in offenen Informationsumgebungen.

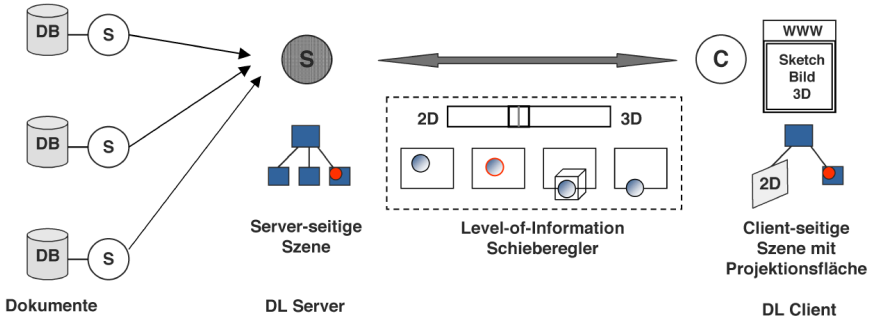

Abbildung 1.3: Das Schieberegler-Prinzip in einer verteilten Visualisierungsumgebung.

1.4 Anwendungsszenario: Kooperative Planungskonferenzen

Ein vielversprechender Einsatzbereich für die entwickelten Konzepte ist die virtuelle Produktdefinition, bei der digitale Prototypen (Digital Mock-Up in der Automobilindustrie) helfen, den Produktenstehungsprozess zu beschleunigen und die Kosten zu verringern. Der enorme Aufwand bezüglich des Informationsaustausches zwischen einem Produzenten und seinen zahlreichen Zulieferern während der Produktentwicklung wird mit den hier vorgestellten Techniken effizienter gestaltet. Gleichzeitig werden die hohen Sicherheitsanforderungen bei der Verarbeitung wertvoller digitaler Produktdaten (*trade secrets*) berücksichtigt und für den Einsatz in der Praxis handhabbar gemacht.

Als Anwendungsszenario für die Arbeit werden kooperative Planugskonferenzen in der virtuellen Produktdefinition gewählt, bei denen mehrere Anwender als Mitarbeiter der beteiligten Firmen gemeinsam mit 3D Dokumenten arbeiten. Hierei werden Anfragen an verteilte Datenbanksysteme gestellt, um ein Retrieval auf den Dokumentbständen durchzuführen, ausgewählte 3D Dokumente werden in der verteilten Umgebung visualisiert, wobei auch Manipulationen an der 3D Daten (z.B. für die Modellierung neuer Produkte) vorgenommen werden, und schließlich werden neu entstandene 3D Dokumente in die digitale Bibliothek integriert. Im Beispiel der virtuellen Produktdefinition arbeiten Konstrukteure eines Automobilherstellers gemeinsam mit den Mitarbeitern von Zuliefererfirmen an der Integration von Einzelkomponenten in einen digitalen Prototypen eines neuen Fahrzeugs. Hierbei werden die 3D Daten, die sich in verschiedenen Datenbanken befinden, den Bauteilen des Prototypen mit Hilfe einer Materialliste (*bill of material*) zugeordnet. Bisher müssen dafür alle Originaldaten vom Zulieferer vor Beginn des Planungsprozesses zum Produkthersteller übertragen werden, wobei einserseits enorme Datenübertragungsmengen erzeugt werden und andererseits kritische Produktdaten eines Zuliefers auf Vertrauensbasis, die eventuell vertraglich abgesichert ist, weitergegeben werden müssen. Die unterschiedlichen verfügbaren Netz- und Rechnerressourcen der Teilnehmer einer solchen Konferenz, die Komplexität der bearbeiteten 3D Daten und schließlich auch die bestehenden Anforderungen an den Schutz der Daten lassen den Einsatz neuer kombinierter Ansätze zur verteilten 3D Visualisierung sinvoll erscheinen. Für die Praxis werden auf diese Weise Arbeitsprozesse in vernetzten Arbeitsumgebungen möglich, die die gseamte Informationsskala von beschreibenden Metadaten bis zu den originalen 3D Modellen nutzen. In Abbildung 1.4 sind solche Arbeitsprozesse für die virtuelle Produktdefinition dargestellt.

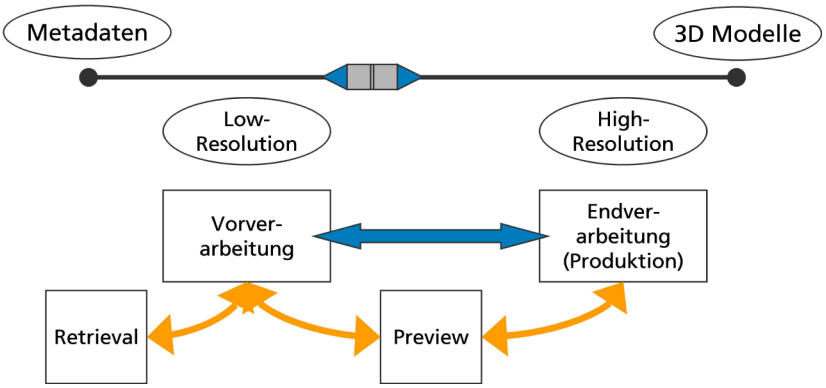

Abbildung 1.4: Arbeitsprozesse nutzen die gesamte Informationsskala für 3D Dokumente.

1.5 Gliederung der Arbeit

Der weitere Aufbau der Arbeit ist wie folgt organisiert. In Kapitel 2 werden die Grundlagen zu den Gebieten Digitale Bibliotheken und 3D Dokumente diskutiert. Die Methoden, die für das Verständnis der Software-Architektur nötig sind, werden dann in Kapitel 3 vorgestellt. In Kapitel 4 wird ein Überblick zum Stand der Forschung auf dem Gebiet der verteilten Visualisierungsumgebungen gegeben. Kapitel 5 stellt das Konzept der Skalierbaren Szenen zur verteilten Visualisierung von 3D Dokumenten vor und beschreibt die darin enthaltenen neuen Ansätze. Der Entwurf der Software-Architektur SCA3D wird in Kapitel 6 dargestellt und ihre Implementierung in Kapitel 7 erläutert. Der Schwerpunkt von Kapitel 8 liegt dann auf der adaptiven Visualisierung in offenen Informationsräumen, wobei das Komplexitäts- und Sicherheitsmanagement der Software-Architektur diskutiert wird. Kapitel 9 beschreibt die Funktionalitäten der Software-Architektur, die das kooperative Arbeiten mehrerer Benutzer mit 3D Dokumenten ermöglichen. Schließlich werden in Kapitel 10 die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf weiterführende Forschungsrichtungen gegeben.

Kapitel 2

Grundbegriffe

In diesem Kapitel werden Grundlagen zu den Themen Digitale Bibliotheken und digitale Dokumente in verteilten Umgebungen beschrieben. Dabei geht es um die Vorstellung des Konzeptes heterogener digitaler Dokumente mit einer Trennung von Struktur und Inhalt, die Einführung von 3D Dokumenten als Untergruppe digitaler Dokumente und den Schutz von Eigentumsrechten von 3D Dokumenten. Außerdem werden Bedingungen für das Arbeiten mit digitalen Dokumenten in verteilten Umgebungen beschrieben.

2.1 Digitale Bibliotheken

Der Begriff der Digitalen Bibliothek (*Digital Library*: DL) entstand im Zuge der rasanten Entwicklung vernetzter Computerumgebungen und der Verarbeitung von Fachwissen in Form von digitalen Dokumenten. Die Digitale Bibliothek ist eine Übertragung der Idee und der Konzepte von herkömmlichen Bibliotheken in den virtuellen und digitalen Bereich. Durch diesen Übergang vom gedruckten Buch an realen Orten zu digitalen Informationen in verteilten, virtuellen Umgebungen wird der Begriff der Bibliothek aber auch stark erweitert und führt zu neuen Möglichkeiten und Herausforderungen. Die Digitale Bibliothek kann als Metapher für die elektronischen Märkte des Wissens und für die Versorgung einer sich entwickelnden Wissensgesellschaft verstanden werden [EF00]. Hierdurch wird klar, wie bedeutend und weitreichend die Möglichkeiten dieser neuen Entwicklung in vielen Bereichen der Gesellschaft sein werden.

Durch folgenden Zitate sollen die Aufgaben einer Digitalen Bibliothek aus verschiedenen Perspektiven beschrieben werden:

- Aus marktwirtschaftlicher Sicht ergibt sich folgendes Bild der Aufgaben [EF00]:

  Es ist die Aufgabe einer digitalen Bibliothek, für einen privaten, akademischen und industriellen Nutzerkreis ... effiziente Dienste anzubieten, die diesem helfen, an das ... in digitalen Dokumenten gespeicherte Fachwissen zu gelangen.

- Aus wissenschaftlicher Sicht stellen sich diese Aufgaben so dar [SSNM97]:

  Es ist die Aufgabe einer digitalen Bibliothek, die Ausnutzung des globalen, vernetzten Informationsuniversums zu verbessern, mit klarer Ausrichtung auf die Bedürfnisse des individuellen Nutzers und seiner Tätigkeit.

- Heutige Bibliotheksbenutzer, die vor allem durch ihren Umgang mit herkömmlichen Bibliotheken geprägt sind, würden die Aufgaben vermutlich folgendermaßen beschreiben [EF00]:

Es ist die primäre Aufgabe einer digitalen Bibliothek, die bekannten Probleme konventioneller Bibliotheken systematisch zu lösen.

Wie aus diesen Zitaten deutlich wird, werden ganz unterschiedliche Anforderungen an eine Digitale Bibliothek gestellt. Die Vorstellungen aus der jeweiligen Sicht erstrecken sich von einer einfachen Verbesserung der herkömmlichen Bibliotheken durch ein digitale Komponente bis hin zu einem sehr umfassenden System für den Zugriff global verteilten Wissens.

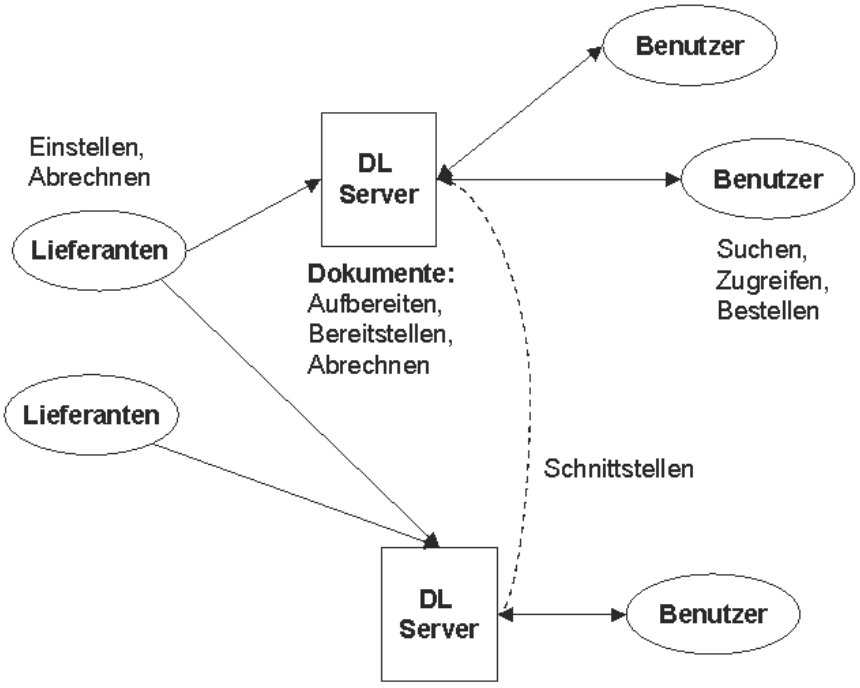

Abbildung 2.1: Skizze eines verteilten DL-Systems mit Lieferanten und Benutzern.

In Abbildung 2.1 ist ein verteiltes System einer Digitalen Bibliothek mit DL-Servern, mehreren Benutzern (privat, wissenschaftlich, industriell) und Lieferanten (Verlage, Fachgesellschaften, Medienunternehmen, Nachrichtenagenturen etc.) dargestellt. Die Benutzer und die Lieferanten sind über verschiedene Wege mit dem zentralen Server der Bibliothek verbunden. Hier wird angenommen, daß sich die Benutzer hauptsächlich über das Internet bzw. ein Intranet mit der Digitalen Bibliothek verbinden, während Lieferanten neben diesen Möglichkeiten auch digitale Langzeitspeicher verwenden, um Dokumente an die Bibliothek zu liefern. Die Hauptfunktionen des DL-Servers sind die Aufbereitung, Bereitstellung und Abrechnung von Dokumenten und Diensten. Daneben sollte aber auch die Betreuung von Benutzern und Lieferanten von einem DL-System unterstützt werden. Die Kernfunktionalität, die Benutzern bereitgestellt werden sollte, wird durch Möglichkeiten für Suche, Zugriff, Bestellung und Abrechnung von digitalen Dokumenten abgedeckt. Lieferanten werden von dem DL-System darin unterstützt, Inhalte einzustellen und diese Lieferung abzurechnen. Wie aus Abbildung 2.1 deutlich wird, muß ein solches System in der Lage sein, mit

mehreren verschiedenen Lieferanten und einer Vielzahl von Benutzern umgehen zu können. Außerdem sind in realen Digitalen Bibliotheken auch Schnittstellen zu anderen DL-Systemen gewünscht, so daß Standards (wie z.B. Dublin-Core [WKL98] oder Z39.50 [Lyn97]) zum Austausch von Dokumenten und Metainformationen nötig sind. Durch die Vernetzung von Digitalen Bibliotheken wird ein System für den Zugriff global verteilten Wissens, wie oben angedeutet, tatsächlich möglich.

In dieser Arbeit werden vor allem folgende Aspekte untersucht, die erst durch den Übergang zur Digitalen Bibliothek ermöglicht werden bzw. berücksichtigt werden müssen:

- Als eine wichtige Erweiterung ergibt sich die Möglichkeit der Kooperation mehrerer verteilter Benutzer mit Hilfe der digitalen Dokumente als gemeinsamer Wissensbasis für ein zu lösendes Problem. Neu entstandene Dokumente können wiederum in die Bibliothek integriert werden, um das enthaltene Wissen in einem Bereich zu erweitern.

- Diese Dokumente können außerdem heterogen sein, d.h. sie umfassen Inhalte mit verschiedenen Datentypen und mehrere Dokumenttypen. Als Beispiel sei ein Dokument über den Bau eines Flughafengebäudes angenommen, das neben dem Text auch Bilder, Video, Audio und 3D Modelle enthält. Die Benutzer eines solchen Dokumentes möchten mit den heterogenen Inhalten arbeiten, wobei gleichzeitig der strukturelle Zusammenhang des Dokumentes, nämlich die verschiedenen Aspekte des Gesamtgebäudes, erhalten bleiben soll. Dies kann nur gewährleistet werden, wenn der Zugriff auf die DL-Dokumente und die lokal genutzten Anwendungen nicht für die verschiedenen Datentypen getrennt behandelt werden müssen. Ein Schwerpunkt der Arbeit ist das Arbeiten mit DL-Dokumenten, die selbst eine 3D Szene darstellen oder eine 3D Datenkomponente enthalten. Diese DL-Dokumente werden im folgenden als 3D Dokumente bezeichnet. Die Komplexität von 3D Dokumenten spielt bei der Visualisierung in verteilten Umgebungen eine entscheidende Rolle. Daher sollen hier Methoden zum Komplexitätsmanagement entwickelt werden.

- Als eine schwierige Herausforderung stellt sich weiterhin der Umgang mit Schutzrechten, wie z.B. den geistigen Eigentumsrechten, von digitalen Informationen in vernetzten Umgebungen heraus. Ein Problem, das im Zuge der Digitalisierung von Wissen allgemein an Bedeutung gewinnt.

2.2 Digitale Dokumente

Ein Dokument stellt Informationen über räumliche oder zeitliche Inhalte bzw. Sachverhalte für den dauerhaften, menschlichen Gebrauch bereit und nutzt ein Trägermedium, auf das die eigentlichen Informationen mit einer bestimmten Syntax und Semantik aufgebracht werden. Dokumente können in analoger oder digitaler Form vorliegen, wobei in beiden Fällen die eigentlichen Inhalte durch eine übergreifende Struktur geordnet werden. Die Digitale Bibliothek arbeitet ausschließlich mit der digitalen Form von Dokumenten, die jedoch durch Digitalisierung von analogen Dokumenten einer bestehenden Bibliothekssammlung erzeugt worden sein können. Zur Einordnung von Dokumenten kann eine Klassifizierung nach der Art des Inhalts dienen [EF00], die folgende Dokumenttypen unterscheidet:

- *Referenzwerke*: Bibliothekskataloge, Produktkataloge, Faktendatenbanken, Gesetzestexte, Lehrbücher etc.

- *Primäre Textveröffentlichungen*: Produktspezifikationen, Forschungsberichte, Monographien, Tageszeitungen etc.

- *Druckbare Nicht-Text-Dokumente*: Bilder, technische Zeichnungen, Karten, Handschriften, wissenschaftliche Meßreihen etc.
- *Nicht-druckbare Dokumente*: Audio-Aufzeichnungen, Filme, Simulationen und Animationen, Multimedia-Produkte.

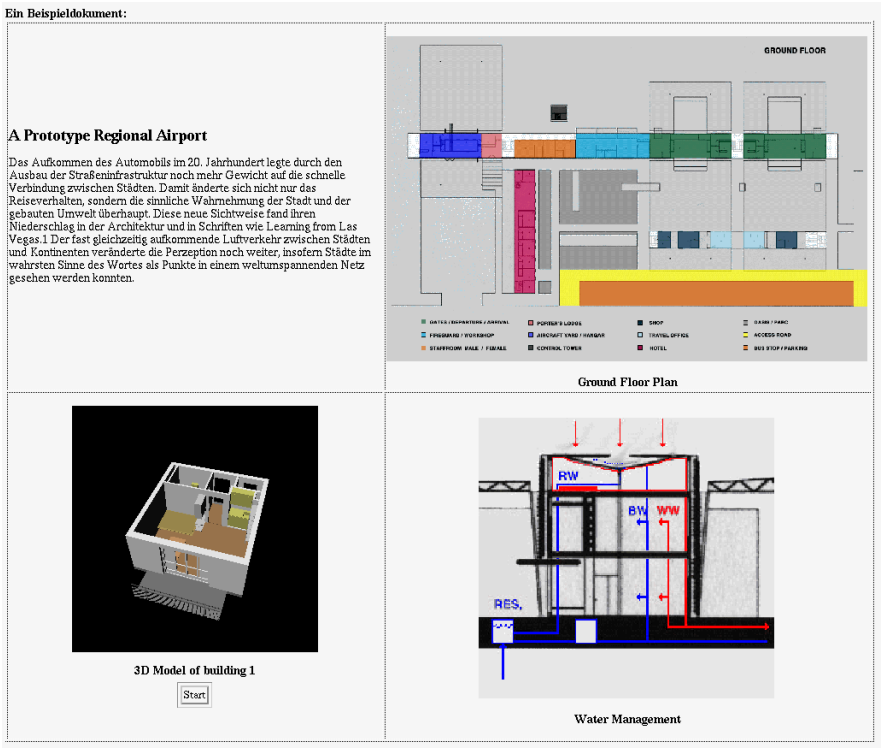

Abbildung 2.2: Beispieldokument Flughafengebäude. Quelle der Einzelbilder und Modelle: ETH Zürich, Fachbereich Architektur.

Die letztgenannte Kategorie von Dokumenten kann durch eine entsprechende Aufarbeitung in druckbare Dokumente umgewandelt werden, z.B. kann aus einer Simulation eine Momentaufnahme eines 3D Modells durch Visualisierung als Bild extrahiert werden oder eine Audio-Aufnahme kann durch Visualisierung des Sprachspektrums dargestellt werden. Diese Konvertierung von Dokumentrepräsentationen wird im folgenden noch eingehender behandelt werden. Zusammengesetze Dokumente, wie sie in Digitalen Bibliotheken möglich werden, enthalten verschiedene Komponenten des jeweiligen Dokumenttyps, wie z.B. das obengenannte Beispieldokument über ein geplantes Flughafengebäude. Hier sind Teile aus Produktkatalogen, eventuell Forschungsberichte über Spezialaspekte des Flughafenbaus, Bilder und technische Zeichnungen und daneben auch 3D Modelle bzw. Filme zu einem Gesamtdokument zusammengeführt. In Abbildung 2.2 ist ein solches Dokumentes mit Hilfe eines entsprechenden DL-Interfaces zur Darstellung auf dem Benutzerbildschirm gezeigt. Für das gesamte Dokument oder eine Komponente kann es notwendig sein, die geistigen Eigentumsrechte oder spezielle Zugriffsrechte zu wahren, wenn Benutzer mit ihnen arbeiten. Verschiedene Ansätze hierfür werden ebenfalls im folgenden beschrieben.

2.2.1 Aufbau digitaler Dokumente

Die Bereitstellung von heterogenen Daten als Dokumente digitaler Bibliotheken und deren Bearbeitung durch Benutzer wird durch die Trennung von Struktur und Inhalt unterstützt. Die mit den Dokumenten gespeicherte Strukturinformation ermöglicht die schnelle Suche von Inhalten, die gezielte Interaktion und auch die Kooperation von mehreren Benutzern mit den digitalen Inhalten. Die Inhalte eines Dokumentes werden hier als Blätter einer Baumstruktur dargestellt, die durch verschiedene Arten von Knoten (Hauptknoten, Gruppenknoten, Ressource-Knoten, Attribut-Knoten etc.) untergliedert ist und so die Struktur des Dokumentes wiedergibt. Teildokumente, Kapitel oder Komponenten eines Dokumentes werden in dieser Baumstruktur durch Gruppenknoten angezeigt. Ressource-Knoten enthalten die Objekte und deren Daten, also den eigentlichen Inhalt. Über eine eindeutige Referenzierung kann auf diese Daten zugegriffen werden. Die Attribut-Knoten weisen jeder Gruppe unterhalb eines Knotens bestimmte Merkmale zu, die den Inhalt genauer charakterisieren, also z.B. den Font und die Fontgröße eines Textes oder die Zugriffsrechte für die Objekte in dieser Gruppe. In Abbildung 2.3 ist der hier beschriebene Aufbau von digitalen Dokumenten gezeigt.

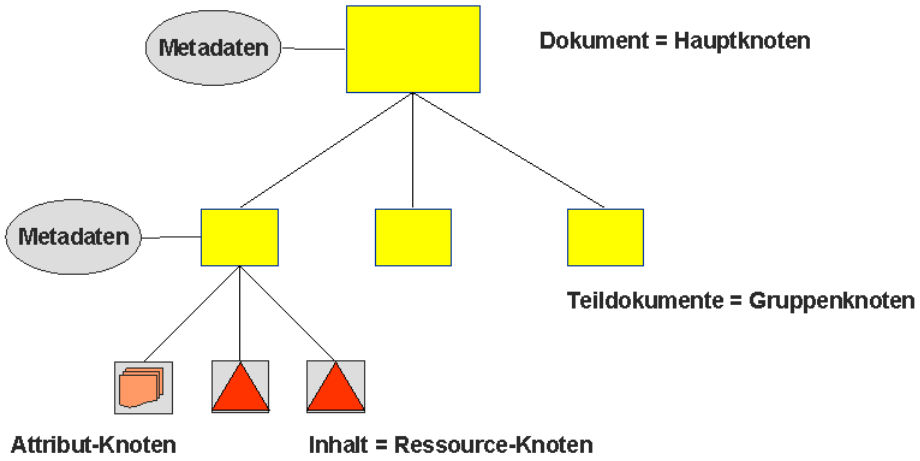

Abbildung 2.3: Trennung von Struktur und Inhalt von DL Dokumenten

Zu jedem Knoten werden Zusatzdaten gespeichert, die das Arbeiten mit den Dokumentinhalten ermöglichen, ohne sofort direkt auf die Daten zugreifen zu müssen. Diese Zusatzinformationen werden als Metadaten bezeichnet (siehe hierzu [Del99]). Wenn ein Benutzer nach bestimmten Dokumenten einer Digitalen Bibliothek suchen möchte, wird er zunächst auf die Metadaten zugreifen, bevor die eigentlichen Daten des Inhalts benutzt werden. Es wird daher beim Arbeiten mit digitalen Dokumenten zwischen metadaten-basierten Methoden und inhalts-basierten Methoden unterschieden. Da metadaten-basierte Methoden sehr viel effizienter im Umgang mit Systemressourcen sind, werden sie für Retrieval und Zugriff von DL-Dokumenten meistens eingesetzt. Inhalts-basierte Methoden können in vielen Fällen genauere Informationen zu einem Dokument liefern und haben auch den Vorteil, für dynamisch sich ändernde Bestände von Digitalen Bibliotheken automatisch Zusatzinformationen generieren zu können.

2.2.2 Eigenschaften heterogener und komplexer Dokumente

Digitale Bibliotheken zeichnen sich unter anderem dadurch aus, daß sie es ermöglichen, mit sehr komplexen und aus mehreren Komponenten zusammengesetzten Informationen umzugehen. Heterogenität und Komplexität als Eigenschaften von DL-Dokumenten sollen hier daher genauer beschrieben werden. Die Klassifizierung in Abschnitt 2.2 hat zwischen mehreren Dokumenttypen unterschieden. Ein Dokument kann außerdem aus mehreren Komponenten mit verschiedenen Datentypen zusammengesetzt werden. Dies führt zu folgender Beschreibung eines **heterogenen Dokuments**:

- Heterogene Dokumente sind aus verschiedenen Dokument- und Datentypen zusammengesetzt. Zur Beschreibung und zum Arbeiten mit solchen Dokumenten werden die zugeordneten Metadaten und enthaltenen Strukturinformationen verwendet. Hierbei wird weiter zwischen einer übergeordneten, einheitlichen Dokumentstruktur und einer spezifischen Komponentenstruktur unterschieden.

Das obige Beispieldokument (Abb. 2.2) enthält eine Dokumentstruktur, die Textkapitel, Überschriften etc, unterscheidbar macht und eine Komponentenstruktur für das integrierte 3D-Modell des Gebäudes in Form des Szenengraphen (hier OpenInventor [Wer98]).

Die Komplexität eines Dokumentes kann nicht in Form eines Zahlenwertes angegeben werden, da dieses Merkmal subjektiv ist. Ein Benutzer, der zum ersten Mal mit einem komplexen Dokument konfrontiert wird, wird erst einmal damit beschäftigt sein, eine Übersicht über Struktur und Inhalt zu gewinnen. Beim weiterem Umgang damit wird sich aber die Komplexität subjektiv weiter reduzieren, da der Benutzer Erfahrung sammelt bzw. übergeordnete Strukturen erkennt. Durch eine abgestufte Änderung des Auflösungsgrades der Wahrnehmung kann ein Benutzer während des Arbeitens mit einem Dokument zwischen einer globalen Navigation in der Struktur und lokaler Interaktion auf inhaltlicher Ebene wechseln. Die folgende Beschreibung von **komplexen Dokumenten** kann ein Kriterium an die Hand geben, wann von Komplexität gesprochen werden kann:

- Die Komplexität eines Dokumentes ist umso größer, je mehr Knoten bzw. Objekte es enthält und je mehr Verknüpfungen zwischen diesen bestehen.

Als Beispiel kann man sich ein digitales Buch mit vielen Referenzen, Fußnoten und Querverweisen vorstellen, oder Verbindungen von 3D Animationen innerhalb eines Dokuments mit bestimmten Stellen in einem Textkapitel. Für ein 3D Dokument, das aus sehr vielen Objekten besteht, steigt die Komplexität mit der Anzahl der Objekte und mit der Menge an Verknüpfungen zwischen ihnen.

2.2.3 Digitale Dokumente in relationalen Datenbanken

Zur strukturierten Speicherung von großen Dokumentkollektionen werden Datenbanken verwendet, die es ermöglichen, alle enthaltenen Dokumentdateien einheitlich zuzugreifen und zu bearbeiten. Durch eine Datenbank (DB) wird garantiert, daß die Daten dauerhaft (persistent) und weitgehend redundanzfrei gespeichert werden. Zur Verwaltung der Dokumentbestände wird eine Datenbank-Management-System (DBMS) verwendet, das ein anwendungsunabhängiges Arbeiten mit den Datenbeständen ermöglicht und die Konsistenz der Datenbank erhält. Unter einem Datenbank-System (DBS) wird meistens die Kombination eines DBMS mit einer oder mehreren Datenbanken verstanden.

Es existieren verschiedene Datenbank-Modelle, die festlegen, wie die Daten intern zusammengefaßt und verwaltet werden [HS00]. Ein Datenbank-Modell ist ein System von Konzepten zur Beschreibung von Datenbanken und legt Syntax und Semantik von Datenbankbeschreibungen fest. Die wichtigsten Modelle sind das relationale Datenbank-Modell, das schon relativ lange entwickelt wird und sich stark durchgesetzt hat, und das objekt-orientierte Datenbank-Modell. Beim relationalen Datenbank-Modell werden die Inhalte in Form von Tabellen dargestellt, für die bestimmte Beziehungen (Relationen) untereinander festgelegt sind. Die Zeilen einer Tabelle entsprechen Tupeln einer Relation und bilden die gespeicherten Datensätze. Die Spalten einer Tabelle werden als Attribute bezeichnet und mit einem Attributnamen versehen.

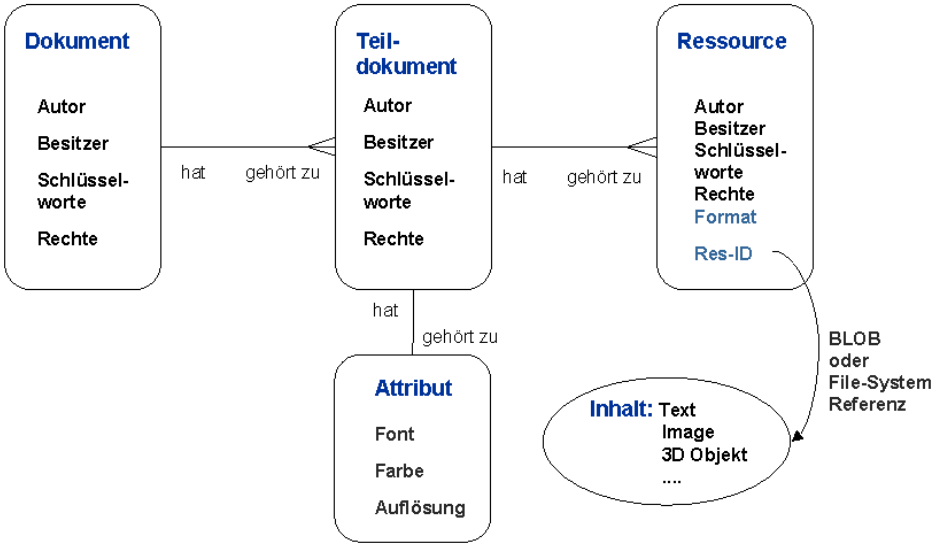

Abbildung 2.4: Entity-Relationship-Diagramm für DL Dokumente in relationalen Datenbanken.

Um mit den Inhalten einer Datenbank arbeiten zu können, sind Grundoperationen wie Mengenoperationen, Projektion, Selektion, Verbund und Umbenennung, auf den Tabellen festgelegt (siehe hierzu [CHRS98]). Eine logisch zusammengehörende Einheit von einer oder mehreren Datenbankoperationen wird als Transaktion bezeichnet. Im Mehrbenutzerbetrieb sorgt das DBMS dafür, daß konkurrierende Transaktionen koordiniert werden und die Datenbank jederzeit in einem konsistenten Zustand ist. Diese Aufgabe wird auch als Synchronisation bezeichnet. Transaktionen werden häufig anhand der *ACID-Eigenschaften* (A-Atomicity, C-Consistency, I-Isolation und D-Durability) charakterisiert.

- *Atomicity*: Transaktionen sind grundsätzlich atomar. Es werden entweder alle Anweisungen einer Transaktion vom DBMS ausgeführt oder keine.
- *Consistency*: Transaktionen überführen eine Datenbank von einem konsistenten Zustand in den nächsten konsistenten Zustand. Inkonsistente Zwischenzustände werden für andere Transaktionen nicht sichtbar.

- *Isolation*: Jede Transaktion läuft in einem vom DBMS simulierten Einbenutzerbetrieb ab.
- *Durability*: Die Ergebnisse einer erfolgreichen Transaktion sind dauerhaft in der DB gespeichert.

Wie im letzten Punkt erkennbar ist, wird zwischen erfolgreich abgeschlossenen (COMMIT) und mit Fehler abgebrochenen (ROLLBACK) Transaktionen unterschieden. Bei einem Rollback wird die Datenbank in den letzten konsistenten Zustand zurückgeführt. Zur Formulierung von Operationen und damit zur Abfrage der Datenbank wurde die Sprache SQL (Structured Query Language) entwickelt. Sie hat sich ebenfalls in hohem Maße durchgesetzt und wird daher von allen Datenbankanbietern unterstützt. Operationen und Transaktionen werden als eine Reihe von SQL-Anweisungen ausgedrückt.

Im weiteren wird nur auf relationale Datenbanken eingegangen, da sie für die Aufgaben im Rahmen der Problemstellung ausreichend und breit verfügbar sind. Zur Beschreibung einer Datenbankanwendung wird ein Datenmodell verwendet, das den Aufbau der Tabellen und die Relationen zwischen ihnen festlegt. Durch ein sogenanntes *Entity-Relationship*-Diagramm kann ein solches Datenmodell dargestellt werden. In Abbildung 2.4 ist die Struktur eines digitalen Dokumentes in Form eines Entity-Relationship-Diagramms gezeigt. Die Knoten der Dokumentstruktur sind hier als Tabellen bzw. Entitäten eines relationalen Datenmodells wiedergegeben. Neben den Metadaten, die durch skalare Werte repräsentiert werden, sind in Tabellenfeldern auch die eigentlichen digitalen Daten als Binärobjekte (BLOB = *Binary Large Object*) angelegt. Als Alternative kann allerdings auch eine Referenz in dem entsprechenden Feld auf das Datenobjekt im einem Filesystem verweisen.

Ein geeignetes Datenmodell in einer relationalen Datenbank sieht übergreifende Dokumentknoten wie Dokumentcluster, Dokument, Teildokument etc., vor und ordnet die Inhalte mit Hilfe von eindeutigen Kennungen den Knoten zu. Ein solches Modell wird z.B. von Borowski et al. [BBH00] beschrieben. Mit Hilfe der Metadaten und der Abfragesprachen SQL können Inhalte einer Dokumentkollektion direkt gesucht, verändert oder extrahiert werden. Die Beziehungen zwischen den Entitäten wird durch Verbindungslinien angegeben. Für ein Dokument gilt hier z.B., daß es mehrere Teildokumente haben kann. Umgekehrt gehört ein Teildokument genau zu einem Dokument. In den Feldern der Tabellen sind die entsprechenden Metadaten für die Entität angegeben. Neben dem Urheber und Schlagworten sind auch Zugriffsrechten und Formate im Beispiel angegeben. Durch diese Strukturierung und die zugehörigen Metadaten ist das gezielte Arbeiten mit den Dokumenteninhalten bei gleichzeitiger Wahrung von Zugriffsbeschränkungen möglich.

Bei den binären Inhaltsobjekten kann es sich um beliebige Datenformate für Audio, Bilder, Video oder 3D Geometrie handeln. Beim Arbeiten mit solchen Inhalten und beim Auslesen können anhand der Metadaten die Informationen über die benötigten und erlaubten Anwendungsmethoden an die jeweiligen Anwendungssysteme übergeben werden. Die Benutzung von heterogen Dokumenten in verteilten Umgebungen kann auf diese Weise organisiert und unterstützt werden.

2.3 3D Dokumente in Digitalen Bibliotheken

Als digitales 3D Dokument wird in dieser Arbeit ein Dokument bezeichnet, das als Ressource-Knoten eine bzw. mehrere 3D Komponenten enthält oder selbst eine vollständige 3D Szene darstellt. Solche 3D Dokumente werden im Prinzip überall verwendet, wo 3D Szenen mit entsprechenden Anwendungen verarbeitet werden. Also z.B. in den Bereichen Ingenieurwesen, Architektur, Design

oder Medizin, die schon seit längerer Zeit 3D Daten in spezialisierter Form mit Computern bearbeitet und genutzt haben. Im Zuge der Ausbreitung digitaler Bibliotheken werden 3D Dokumente aber in sehr viel allgemeinerer Weise von einem breiten Anwenderkreis verwendet werden. Gleichzeitig werden die zugänglichen 3D Informationen immer komplexer. Diese neue Nutzungsform erfordert Verfahren zur Handhabung großer 3D Szenen in verteilten Informationsräumen, wobei Kombinationen von bekannten und neuen Ansätzen den breiten Anforderungen am besten gerecht werden können. Die in dieser Arbeit beschriebene und untersuchte Software-Architektur stellt einen solchen neuen Ansatz dar, bei dem lokale und Remote-Visualisierung von 3D Dokumenten dynamisch kombiniert werden, um den Aspekten Sicherheit, Netzanbindung und lokaler Client-Performanz in heterogenen Umgebungen gerecht zu werden.

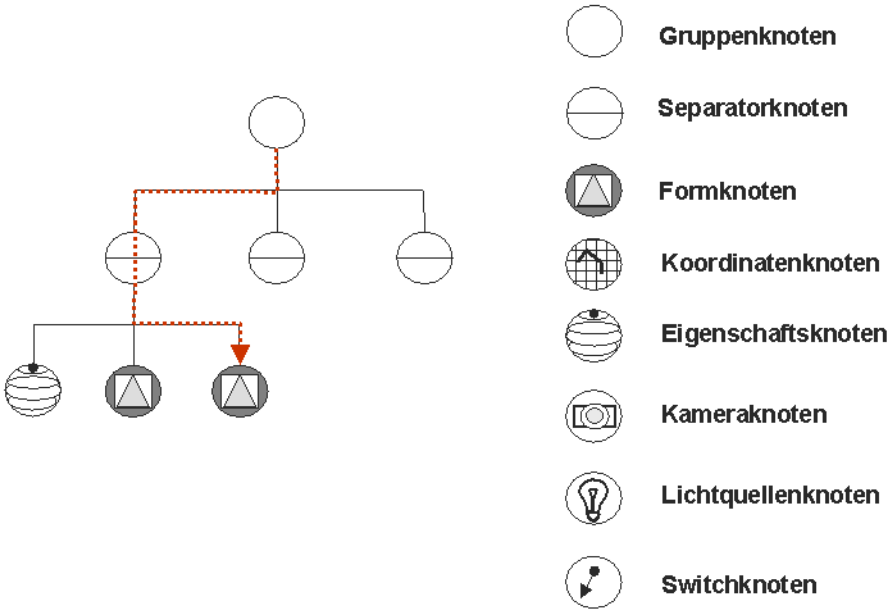

Abbildung 2.5: Szenengraph-Struktur mit Pfadinformation in OpenInventor

Wie die allgemeinen digitalen Dokumente, die oben beschrieben wurden, enthalten auch 3D Dokumente Strukturinformation, Inhaltsobjekte, Attributbeschreibungen und eventuell Metainformationen. 3D Dokumente werden hier daher als Untergruppe von allgemeinen digitalen Dokumenten beschrieben. Im Fachgebiet der Computergraphik wurden in den letzten Jahren verschiedene Konzepte und Formate zur Beschreibung, Bearbeitung und Visualisierung von 3D Szenen entwickelt. 3D Szenen bestehen im Prinzip aus den 3D Objekten, einer oder mehreren Lichtquellen und zumindest einer Kameraeinstellung, d.h. Blickpunkt- und Winkel. Die Strukturinformation von 3D Szenen wird heute meistens in Form einer Baumstruktur, dem sogenannten Szenengraphen beschrieben, der mathematisch einen gerichteten, azyklischen Graph darstellt. An den Verzweigungen enthält der Szenengraph wiederum verschiedene Knotenarten zur Gruppierung von Objekten und Untergraphen. Durch Angabe eines Pfades in dem Szenengraph kann ein Objekt eindeutig referenziert werden und sogar zur Weitergabe gespeichert werden. Jeder Knoten entspricht einem festgelegten Typ, z.B. Gruppenknoten, Formknoten, Materialknoten etc., und enthält eine Anzahl von

Feldern mit variablen Werten, die zur Beschreibung der Parameter der 3D Szenen benutzt werden. In Abbildung 2.5 ist ein Szenengraph dargestellt, wie er von der OpenInventor Visualisierungs-Software verwendet wird. Der Vergleich mit Abbildung 2.3 zeigt, daß beide Darstellungsarten ein Dokument mit ähnlichem Aufbau wiedergeben.

Die geometrischen Inhalte werden durch die Formknoten (OpenInventor: Shape Nodes) im Szenengraphen repräsentiert. In den Feldern der Formknoten sind die eigentlichen geometrischen Daten abgelegt, z.B. der Radius und Mittelpunkt einer Kugel bei geometrischen Primitiven oder Koordinaten eines Dreiecks bei 3D Maschennetzen. Die wichtigsten Beschreibungsarten von Geometrieinformation sind heute Polygon- bzw. Dreiecksnetze, implizite Funktionsdarstellungen geometrischer Primitive (Kugel, Quader, Konus etc.), parametrisierte Kurven bzw. Flächen (Spline, Bezier-Kurven etc.) [FDFH90] und Oberflächen aus zusammengesetzten Elementen parametrisierter Flächen (*Subdivision-Surfaces*) [Loop87]. Polygonale 3D Netze, die vielfach Verwendung finden, haben den Nachteil, daß sie nicht mathematisch exakt und damit auch nicht parametrisierbar sind. Für 3D Dokumente in verteilten Umgebungen bedeutet dies, daß die eigentliche Modellierinformation in einem sehr frühen Stadium verloren geht und daher nicht in der weiteren Verwendungskette der 3D Szenen genutzt werden kann. Außerdem wird der Umgang mit Komplexität durch die frühe Erzeugung von 3D Netzen im Tesselierungsschritt erschwert. Hier wird daher versucht, die Einschränkung auf 3D Netze zu vermeiden und stattdessen mit den Repräsentationen von 3D Objekten zu arbeiten, die sich für die gegebenen Anforderungen am besten eignen. Auf verschiedene Repräsentationen von Objekten wird im folgenden Kapitel 3 genauer eingegangen.

Zur vollständigen Beschreibung von 3D Dokumenten werden weiterhin Materialeigenschaften benötigt, die Angaben über die Oberflächeneigenschaften, wie z.B. Farbe oder Reflexion, von Objekten machen. Die Materialeigenschaften sind im Szenengraphen als Attributknoten und durch deren Felder wiedergegeben. Zur einfacheren Handhabung werden häufig Texturen zur Modellierung der Objektoberfläche benutzt, d.h. es werden Bilder auf die geometrische Form projiziert. Um die Visualisierung zu unterstützen, sind in 3D Dokumenten auch Lichtquellen und Kameraknoten mit Blickwinkel- und positionen enthalten. Metainformationen, wie z.B. Objektnamen oder Rechteinformationen, können durch spezielle Felder mit den Knoten verknüpft werden, so daß sie bei der Verarbeitung genutzt werden können.

Bei der interaktiven Visualisierung von 3D Dokumenten laufen die folgenden Schritte ab. Sobald die 3D Daten in eine geeignete Anwendung geladen werden, wird die Strukturinformation zum Aufbau eines Szenengraphen in Form einer internen Szenendatenbank verwendet. Zur Visualisierung der Szene wird der Szenengraph von oben nach unten vollständig durchgegangen. Dabei werden Aktionen auf den Szenengraphen angewendet. Beispiele für Aktionen, die dabei auf die Szene oder Teilszenen angewendet werden können, sind Rendering, Picking und die Berechnung von Bounding-Box-Information. Die Bounding-Box eines 3D Objektes ist ein Quader mit minimalem Volumen, der das Objekt vollständig umschließt. Verschiedene moderne Graphik-APIs nutzen solche Szenengraphen, um die Entwicklung von interaktiven Visualisierungsanwendungen zu unterstützen. Oft verwendete Formate zur Beschreibung von 3D Dokumenten sind das OpenInventor-Format *IV* [Wer98], das *VRML*-Format [BK96], das speziell für die Verwendung von 3D Daten im WWW entwickelt wurde, oder das OBJ-Format. Es existieren jedoch noch zahlreiche andere Formate, die hier nicht aufgeführt werden sollen.

2.4 Sicherheitsaspekte von digitalen 3D Dokumenten

Digitale Bibliotheken erlauben den Zugriff auf alle Formen digitaler Dokumente in verteilten Informationsräumen, d.h. in Systemen von Computern, die durch ein lokales oder durch Telekommunikationsnetze ständig oder nur zeitweise verbunden sind. Die heute wichtigste und verbreitetste Form eines verteilten Informationsraumes ist aufbauend auf dem Internet das *World-Wide-Web* (WWW). In Kombination mit lokalen Intra-Netzen wird das WWW zu einer sehr flexiblen Basis für Digitale Bibliotheken, da die Verfügbarkeit des angebotenen Wissens lokal und global stark anwächst. Die breite Verfügbarkeit von digitalen Dokumenten wirft aber auch Probleme auf, die für ein lokales Angebot von analogen Informationen, wie Büchern, Bilddrucken oder Tonträgern in herkömmlichen Bibliotheken, nicht in einem ähnlichen Maß entscheidend waren. Durch den Übergang von analogen zu digitalen Medien tritt das Problem auf, daß Informationen beliebig oft ohne Qualitätsverlust kopiert werden können. Bei analogen Medien treten bei wiederholtem Kopieren Generationseffekte auf, die hohe qualitative Einbußen gegenüber der Originalqualität schon nach einigen wenigen Vervielfältigungsschritten bedeuten. Dies hatte den Effekt, daß geistige Eigentumsrechte einfacher geschützt werden konnten und *Copyright*-Verletzungen seltener waren. Das Original eines analogen Dokumentes mit hoher Qualität blieb immer beim Besitzer, solange er es nicht unter festgesetzten Bedingungen weitergab. Für digitale Medien kann dieser Effekt nicht genutzt werden, um das geistige Eigentumsrecht real zu schützen. Sobald eine Kopie angefertigt und verteilt wird, existiert ein identisches Replikat der Daten, das beliebig oft und relativ einfach weitergegeben werden kann. Im Zusammenhang mit der Gewährleistung von Urheberrechten sind weiterhin auch die Zugriffskontrolle auf Daten und deren Integrität in verteilten Informationsräumen in besonderem Maß zu beachten. Beispiele für solche schützenswerten Daten sind technische Dokumente im Ingenieurbereich oder personenbezogene Daten in der Medizin, aber auch künstlerische Arbeiten, wie virtuelle Welten oder Musikaufnahmen.

Sicherheitsanforderungen in verteilten Informationsräumen

Diese Probleme führten zur Entwicklung verschiedener Verfahren zum Schutz von digitalen Dokumenten in der Informationstechnik. Die wichtigsten Sicherheitsanforderungen, die hierbei berücksichtigt werden, sind im folgenden aufgeführt und genauer beschrieben [Dit00]:

- *Urheberrechte*: Schutz von geistigem Eigentum, Eigentümer- und Benutzerkennzeichnung, Vermeidung und Aufdeckung von Copyright-Verletzung
- *Zugriffskontrolle*: Wirksame Methoden zur Kontrolle des Systemzugangs und Zugriffsbeschränkungen auf Systemfunktionen und Datenbanken.
- *Integrität*: Erkennung von Abweichungen von den Originaldaten verursacht durch Manipulation oder Übertragungsfehler. Methoden zur Überprüfung.
- *Authentizität*: Möglichkeiten zur Identifikation von Sendern, Empfängern und Dokumenten, d.h. Authentifizierung von Kommunikation und Daten.
- *Vertraulichkeit*: Schutz vor Zugriff auf vertrauliche Daten durch Unbefugte auf dem Server oder bei der Übertragung.

2.4.1 Verfahren zur Berücksichtigung der Sicherheitsaspekte

Verfahren zur Realisierung der genannten Anforderungen werden hauptsächlich im Fachgebiet der Kryptologie untersucht und weiterentwickelt. Dabei unterteilt man dieses Gebiet in die Kryptographie, Kryptoanalyse und Steganographie. Die Kryptographie beschäftigt sich mit Methoden zur Transformation von Daten durch Verschlüsselung zur Authentifizierung, Identifizierung und Erstellung digitaler Signaturen. Die Kryptoanalyse untersucht Verfahren zur Rücktransformation der Daten und damit auch Angriffe auf die kryptographischen Verfahren. Die Steganographie schließlich entwickelt Methoden zur verdeckten Kommunikation von Daten, so daß unbefugte Dritte von einem Austausch der geschützten Daten keine Kenntnis bekommen können. Als Beispiel sei die geheime Übertragung von Informationen in Bilddatenströmen genannt, bei der die Redundanz und Irrelevanz der visuellen Trägerdaten genutzt wird, um die eigentliche Kommunikation in den Bildern zu verstecken.

Neben der Verschlüsselung ist ein weiteres Verfahren zum Schutz von Urheberrechten und darüberhinaus zur Echtheitsprüfung die Markierung der geschützten Daten durch Wasserzeichen, wie es seit langem im analogen Bereich, z.B. bei Geldscheinen, erfolgreich praktiziert wird. Digitale Wasserzeichen werden als zusätzliche Daten in das Originaldokument eingebracht und möglichst robust mit den Daten verknüpft. Das Watermarking hat sich aus der Steganographie entwickelt und ist heute ein wichtiges eigenes Forschungsgebiet. Wasserzeichen können sichtbar in ein Dokument eingebracht werden, damit eine Identifizierung der Daten jederzeit visuell geschehen kann, oder nicht-wahrnehmbar in den Daten versteckt werden. Für nicht-wahrnehmbare Wasserzeichen unterscheidet man die Einbettung der Watermark-Nachrichten in ein Trägerdokument (*Cover, Carrier* oder auch Original) und den Abfrageprozeß (*Watermark retrieval*) [Dit00]. Bei der Abfrage wird mit einem geheimen Schlüssel das Trägerdokument so verarbeitet, daß die Watermark-Nachricht ausgegeben und als richtig erkannt werden kann. Dieser Schritt wird zum Nachweis der Urheberschaft und zur Überprüfung bzw. zum Auslesen von zusätzlichen Informationen, wie Kundendaten, Kopienummer etc., verwendet.

2.4.2 Mängel der heutigen Verfahren

Die genannten Verfahren können generell auf digitale Daten angewendet werden, jedoch erfordern die verschiedenen Datentypen, wie Text, Bilder, Video, Audio und 3D Szenen, eine spezielle Anpassung der Methoden. Beim Watermarking werden in den meisten Fällen die Komponenten von zusammengesetzten Dokumenten mit eigenen Wasserzeichen versehen, die untereinander keine Verknüpfung besitzen. Dadurch wird der Schutz von heterogenen Dokumenten erschwert. Für 3D Szenen existieren im Bereich Watermarking fast ausschließlich Verfahren, die auf polygonalen 3D Netzen arbeiten und die Manipulation der Netzmaschen zur Einbringung des Wasserzeichens verwenden. Zur Kennzeichnung von anderen Repräsentationsformen von 3D Szenen sind wenige Methoden bekannt. Ein allgemein anwendbares und robustes Verfahren zum Schutz von 3D Dokumenten existiert heute nicht, so daß ein Konzept zum Management von Sicherheitsanforderungen über reine kryptographische Verfahren hinausgehen muß. Diese Sichtweise wird auch von Bissel et al. [Bis00] nach einer Evaluation existierenden Verfahren zum Schutz durch Wasserzeichen für digitalen Ausstellungsobjekten von Museen bestätigt.

2.4.3 Skizze eines umfassenden Management-Szenarios für Schutzrechte von 3D Dokumenten

Zur Konzeption eines umfassenden Management-Szenarios sollten folgende Schritte berücksichtigt werden: Authentifizierung von Kunde und Anbieter, Kommunikation von bestehenden Rechten, Verhandlung von neuen Rechten und Austausch von Dokumentrepräsentationen, die an den aktuellen Stand der Rechtesituation angepaßt werden können. Dokumentrepräsentationen können z.B. durch Einbettung von Wasserzeichen, Anhängen von Signaturen, Transformation in eine andere Repräsentationsform oder durch direkte Kopie erzeugt werden. Ein Kunde, der keine Rechte zur Kopie eines Originaldokuments erworben hat, erhält ein DL-Dokument in einer niedrigen Qualität bzw. in einer Repräsentationsform mit geringem Informationsgehalt. Erwirbt er mehr Rechte bis hin zum Kauf des Originals, bekommt er schrittweise immer mehr Information über das Dokument. Eine Verschlüsselung der übertragenen Daten mit symmetrischen oder asymmetrischen Verfahren kann das Gesamtkonzept vervollständigen. In Abbildung 2.6 ist ein Schichtenmodell für ein Sicherheitskonzept gezeigt, das prinzipiell vom Aufbau des *Multimedia-secure Gateway* von Nahrstedt und Qiao [NQ98] abgeleitet ist.

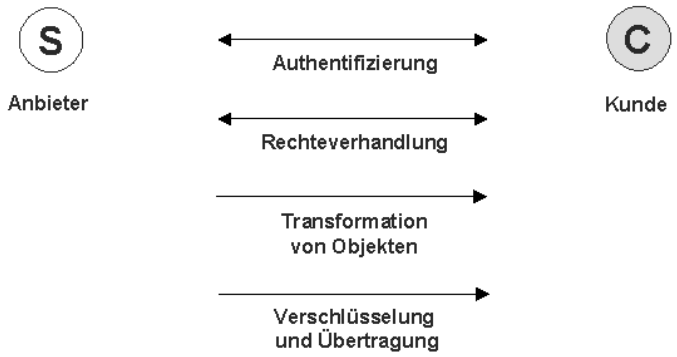

Abbildung 2.6: Schichten eines Management-Szenarios.

Um ein solches Szenario umsetzen zu können, wird ein internes Dokumentmodell benötigt, das Möglichkeiten zur Integration, Evaluation und Extraktion von sicherheitsrelevanten Daten bietet. Ein internes Dokumentmodell wird beim Einlesen des Dokumentes, z.B. aus einer Datenbank in eine Anwendung, generiert und folgt festgelegten Regeln. Hierfür eignet sich besonders ein objekt-orientiertes Dokumentmodell, bei dem alle Bestandteile und das Dokument selbst die Eigenschaften von Objekten im objekt-orientierten Sinn haben: die eigentliche Objektdaten werden gekapselt, es existiert eine eindeutige Identifizierung für jedes Objekt und Zugriffsmethoden sind in einer wohl definierten Schnittstelle festgelegt. Ein Beispiel ist z.B. das *Document Object Model: DOM* des *World Wide Web Consortiums* W3C [DOM00].

In Abbildung 2.7 ist ein Dokumentmodell gezeigt, das die beschriebenen Eigenschaften für 3D Dokumente umsetzen soll. Ein Dokument enthält ein oder mehrere Objekte, die als Knoten einer Baumstruktur definiert sind. Die eigentlichen Objektdaten sind ebenfalls Knoten des jeweiligen Datentyps. Die Objekte sind im Dokument gekapselt und durch eine Schnittstelle von Methoden zugänglich gemacht. Diese Schnittstelle umfaßt im wesentlichen Methoden für Erzeugung, Retrie-

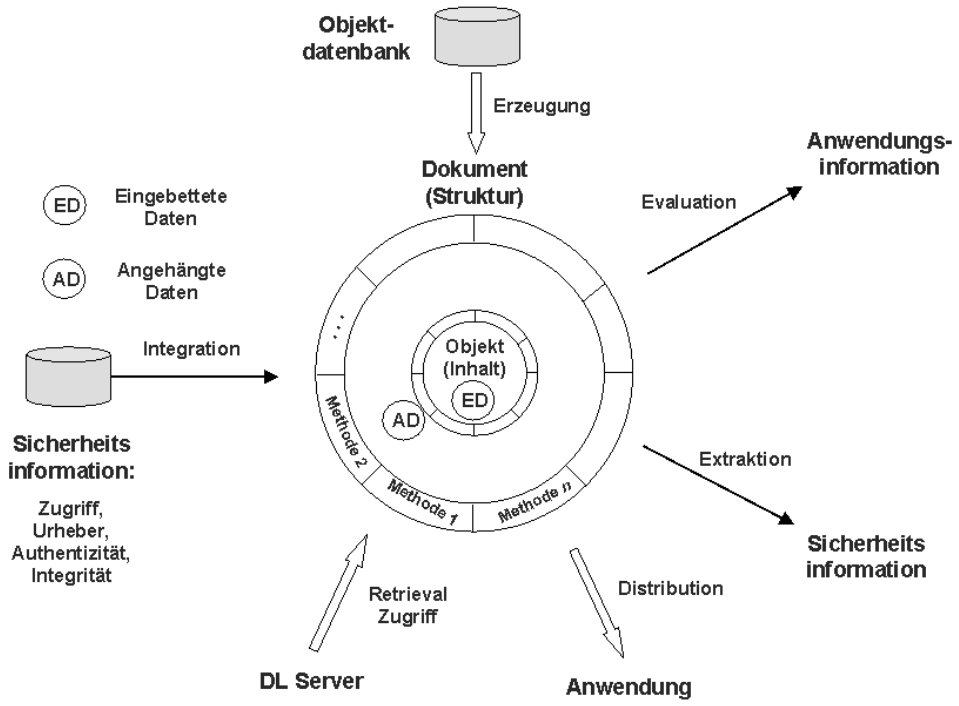

Abbildung 2.7: Objektmodell zur Integration, Evaluation und Extraktion von Sicherheitinformationen für digitale Dokumente.

val und Verteilung von Dokumenten. Außerdem werden die Methoden zur Integration, Extraktion und Evaluation von Sicherheitsinformationen festgelegt.

Die Sicherheitsinformationen, die in das Dokument integriert werden sollen, beziehen sich auf die Zugriffskontrolle, die Urheberschaft, die Authentizität und die Integrität von Objektdaten. Die kryptographischen Verfahren, die oben diskutiert wurden, funktionieren, indem zusätzliche Daten an ein Objekt angehängt (Digitale Signatur) werden oder durch Einbettung der Daten in das Objekt selbst (Watermarking). Es müssen daher Methoden zum Anhängen und Einbetten dieser sicherheitsrelevanten Daten für den jeweiligen Dokument- und Datentyp gefunden werden. Neben der Integration von Daten sind auch Methoden zur Evaluation und Extraktion (vgl. Abbildung 2.7) nötig, um in einer Anwendung Zugriff auf die sicherheitsrelevanten Daten zu erlauben. Um das Arbeiten mit einem Dokument zu optimieren, stehen im Dokument selbst verschiedene Indizes zur Verfügung, die Verknüpfungen mit den Objekten enthalten. Ein Index für die Zugriffskontrolle ermöglicht den Abgleich mit den Zugriffsrechten eines Benutzers, der mit bestimmten Objektrepräsentationen arbeiten möchte.

2.4.4 Repräsentation und Integration von Sicherheitsinformationen

In diesem Abschnitt sollen die Methoden zur Integration, Extraktion und Evaluation von sicherheitsrelevanten Zusatzinformationen während der Bearbeitung von 3D Dokumenten diskutiert wer-

den, die in Abschnitt 2.4.3 für allgemeine Dokumente beschrieben wurden. Die für ein Sicherheitsmangement benötigte Funktionalität umfaßt die Urheberidentifizierung, Kundenidentifizierung, Kopienkontrolle bzw. den Schutz des Originals, die Nachweisbarkeit der Unversehrtheit und Vertraulichkeit durch Zugriffskontrolle. Für die Realisierung dieser Aufgaben werden zusätzliche Daten an die Objektdaten eines Dokumentes angehängt oder darin eingebettet, meist in der Form von Text (1D) oder Mustern (2D, 3D).

Die einfachste Form von Informationen zum Schutz des Urheberrechts ist eine Textkette mit den Metadaten des Objektes, die z.B. im Dublin- Core-Format vorliegen können, also z.B. Informationen über den Autor, das Entstehungsdatum und den Eigentümer. Der sogenannte Dublin-Core wird in der Praxis oft verwendet und legt einen Satz von Attributen zur Beschreibung von Dokumenten durch Metadaten fest (siehe [WKL98]). Das folgende Beispiel repräsentiert ein 3D Dokument bzw. Objekt, das durch seine Metadaten im Dublin-Core Format identifiziert werden kann:

„ **Metadaten:** ID , Author, Entstehungsdatum, Eigentümer, Copyright **::**
Document-Distribution-Class: Klasse B (Kopie als Dreiecksnetz erlaubt) **:: Auflösung:** Klasse B-Wert δ“

Neben den Metadaten sind Textdaten eingefügt, die Aufschluß über die Bedingungen bei der Verteilung von Kopien geben. Es kann eine Klasseneinteilung für Objekte vorgenommen werden, die diese Bedingungen wiedergibt, also z.B. *Klasse A* für Objekte, die als Original verteilt werden dürfen, und *Klasse B* für solche, die nur als Dreiecksnetz weitergegeben werden dürfen. Als weitere Information wird ein Wert eingefügt, der Aufschluß über die maximale Auflösung von Objektkopien angibt. Im Fall von Netzrepräsentationen also die Größe der Dreiecksmaschen. Mit diesen Informationen, die als Text oder Signatur an eine Objekt angehängt werden, oder in codierter Form als Wasserzeichenmuster in die Objektdaten eingebracht werden, kann die oben genannte Funktionalität weitgehend realisiert werden.

Die gängigsten Techniken zur Integration arbeiten mit Watermarking, inhalts-basierten Signaturen ([DSS99]) und Verschlüsselung. In Tabelle 2.1 wird eine kurze Beschreibung der verschiedenen Methoden mit ihren Eigenschaften gegeben.

Beim Watermarking werden Muster oder in Muster codierte Texte eingebracht. Man unterscheidet: wahrnehmbare bzw. nicht wahrnehmbare, blinde bzw. nicht-blinde und schließlich robuste bzw. fragile Wasserzeichen (siehe [KP00]). Die Bezeichnung nicht blind bzw. blind bezieht sich darauf, ob zur Rekonstruktion des Wasserzeichens die Originaldaten benötigt werden oder nicht. Robust bedeutet hier Beständigkeit gegenüber Manipulationen der Originaldaten. Fragile Wasserzeichen sollen gerade im Fall bestimmter Manipulationen zerstört werden. Im Fall der Verschlüsselung werden die Schlüsselinformationen in einer sicheren Form zur Verfügung gestellt. Man unterscheidet symmetrische Verfahren, bei denen der gleiche Schlüssel für Codierung und Decodierung verwendet wird, und assymmetrische Verfahren, die mit einem öffentlichen Schlüssel zur Codierung und einem privaten zur Decodierung arbeiten.

Die Einbettung von Wasserzeichen in 3D Modelle ist ein aktuelles Forschungsgebiet. Die meisten der bekannten Verfahren arbeiten auf den Geometrieelementen von 3D Netzen. Dadurch wird der Einsatz von Wasserzeichen stark eingeschränkt, da 3D Objekte bestehend aus parametrisierten Oberflächen nicht direkt markiert werden können. Erst nach dem Tesselierungsschritt können auch hier Wasserzeichen verwendet werden. Verfahren zum Einbringen von digitalen Informationen in polygonale 3D Modelle werden z.B. von Ohbuchi [OMA98] und Benedens [Ben99] vorgeschlagen. Ein Verfahren zur Markierung von Parametern von Gesichtsanimationen im MPEG-4 Format wird von Hartung et al. [HEG98] besprochen. Die Verfahren nutzen im wesentlichen die folgenden drei Ansätze zur Einbettung von Wasserzeichen in polygonale Modelle:

	Wasserzeichen	**Signaturen**	**Schlüssel**
Beschreibung	Einbettung von Mustern oder codierten Texten in Objektdaten	Extraktion von inhalts -basierten Merkmalen, Erzeugung der Signatur durch Lauflängencodierung, plus Verschlüsselung	Information zur Rücktransformation mit einem gegebenen Verschlüsselungs-algoritmus
Urheber-identifizierung	Urheber-identifizierende Muster oder Texte, unsichtbar	Anhängen der verschlüsselten Signatur, plus *Public-Private-Key* Verfahren (PPK)	In Kombination mit Signatur
Kunden-identifizierung	*Fingerprints*: unsichtbare Kunden-identifizierende Muster oder Texte	Anhängen der verschlüsselten Signatur plus PPK-Verfahren	In Kombination mit Signatur
Kopien-kontrolle	Oft Blackbox-Verfahren zur öffentlichen Verifi-, zierung, unsichtbar, blind, Muster oder Texte	-	Durch Verwendung lizensierter Hardware Teilschutz möglich
Unversehrt-heits-nachweis	Unsichtbare fragile Wasserzeichen, Mustereinbettung	Anhängen der codierten Signatur, Verwendung von Merk-malen, die unzulässige Manipulationen anzeigen	In Kombination mit Signatur
Vertraulich-keits-schutz	-	Zugriffskontrolle durch feste Kundensignaturen	Schutz durch assymetrische Ver-fahren oder große Schlüssellänge

Tabelle 2.1: Gebräuchliche Repräsentationsformen von Sicherheitsinformationen.

- Modifikation der Vertex-Koordinaten
- Modifikation der Vertex-Topologie
- Kombination von Geometrie- und Topologie-Modifikation

Alle Verfahren haben das Ziel robust gegen bestimmte Geometrieoperationen, wie allgemeine Transformationen, Ausschnittsbildung bzw. lokale Deformation oder topologische Modifikation zu sein. Eine Diskussion der angesprochenen Verfahren hinsichtlich ihrer Robustheit wird in [Dit00] durchgeführt.

Am Beispiel der *Digital Versatile Disc* (DVD) sollen Beispiele für kombinierte Verfahren zur Durchsetzung von Urheberechten gegeben werden. Da DVD Audio- und Videodaten mit sehr hoher Qualität zur Verfügung stellt, haben Urheber, Produzenten und Verleger ein großes Interesse an der Wahrung des Copyrights. Es sind unter anderem zwei wichtige Schutzmechanismen vorgesehen [MCB98]:

- *Content Scrambling System*: Beim CSS werden Videodaten im MPEG-2 Format auf einer DVD abgelegt. Das für die Rücktransformation benötigte Schlüsselpaar wird im *lead-in* Bereich der DVD abgelegt und kann bei einer Vervielfältigung nicht mitkopiert werden. Nur DVD-genormte Geräte können die Daten abspielen, wodurch ihre Verbreitung vorangetrieben wird.

- *Copy Generation Management System*: Mit Zusatzinformationen im MPEG-Header wird geregelt, ob die Daten kopiert werden dürfen. Es werden die Zustände: Kopierschutz, ohne Kopierschutz und Einmalkopie erlaubt. Durch Verwendung von Abspielgeräten, die diese Information übergehen, kann das System umgangen werden. Es ist also nur für DVD-genormte Geräte sinnvoll und baut darauf, daß diese sich durchsetzen.

Wie schon an dieser Beschreibung einiger Methoden zum Sicherheitsschutz der DVD zu erkennen ist, gibt es derzeit kein wirklich sicheres Verfahren, um Urheberrechtsschutz wirkungsvoll durchzusetzen. Vielmehr können alle Verfahren mit gewissen Aufwand unwirksam gemacht werden. Für den digitalen Bereich bedeutet dies, daß der Schutz des Originals große Bedeutung hat. Die Verbreitung von Kopien mit einer teilweise verminderten Qualität oder von transformierten Repräsentationsformen stellt daher eine wichtige Möglichkeit zur Wahrung von Urheberinteressen dar. Mit Hilfe eines objekt-orientierten Dokumentmodells, das die Objektdaten eines Dokumentes kapselt und eine Methodenschnittstelle zum kontrollierten Zugriff bereitstellt, und durch zusätzliche Verschlüsselung von Daten kann ein solches System zum sicheren Management von 3D Dokumenten realisiert werden.

2.5 Arbeiten mit digitalen Dokumenten in verteilten Umgebungen

Das in Abschnitt 2.1 beschriebene *Digital-Library*-Szenario zeigt, daß ein Konzept zum Arbeiten mit DL-Dokumenten in hohem Maße Techniken aus dem Gebiet der verteilten Systeme nutzt. In diesem Abschnitt sollen die wichtigsten Techniken verteilten Arbeitens mit DL-Dokumenten und die Anforderungen besprochen werden, die an eine verteilte Architektur gestellt werden. Eine Anwendung, die Möglichkeiten anbietet, mit verteilten Prozessen, verteilten Daten und verteilten Benutzern umgehen zu können, wird als verteiltes System bezeichnet. Eine Digitale Bibliothek, die über ein Datennetz Dokumente und Dienste für global verteilte Benutzer anbietet, setzt in diesem Sinn auch eine verteilte Architektur zur Bewältigung ihrer Aufgaben voraus.

Verteiltes Arbeiten: Retrieval, interaktive Visualisierung und Kommunikation

Um die Dienste einer Digitalen Bibliothek zu nutzen, d.h. mit den enthaltenen Dokumenten arbeiten zu können, müssen Möglichkeiten zum Retrieval der Dokumente, zum Zugreifen und zum Bestellen bzw. Erwerb realisiert werden. Für interaktive und kooperative Aufgaben werden auch Möglichkeiten zur Manipulation von Dokumentrepräsentationen und zur Kommunikation gebraucht. Die letzteren Aspekte sind vor allem wichtig, wenn es darum geht, mit dem DL-Server selbst und mit anderen Benutzern synchronisiert zu arbeiten, um ein Rechte-Management und kooperatives Arbeiten für geschützte Dokumente zu erlauben. Diese Möglichkeiten gehen über reines Arbeiten mit den Metadaten einer Dokumentkollektion hinaus, für das z.B. eine Kommunikation mittels des HTTP-Protokolls [RFC2616] ausreichen würde.

Für das verteilte Arbeiten mit 3D Dokumenten sollen die drei Hauptaspekte noch einmal beschrieben werden:

- *Retrieval*: Um in großen Kollektionen von DL-Dokumenten 3D Szenen und Objekte abfragen und für weitere Arbeitsschritte aussuchen zu können, existieren Methoden zur Abfrage von Datenbanken in verteilten Informationsräumen. In den meisten Fällen werden für das Retrieval Metadaten benutzt, die nachträglich für ein Dokument annotiert wurden. Über eine WWW-Schnittstelle und die Abfragesprache SQL kann ein einfaches Retrievalsystem einer Digitale Bibliothek realisiert werden. Um auf den Inhalten von Dokumenten arbeiten zu

können, werden inhalts-basierte Verfahren, wie automatische Merkmalsextraktion für Bilddaten oder *Fulltext Retrieval* für Text, eingesetzt (siehe hierzu [Del99]). Im Falle von 3D Dokumenten werden hauptsächlich geometrische Merkmale aus den Daten extrahiert und zur Abfrage benutzt [PR99]. Eine Kombination von inhalts-basierten Methoden zum 3D Retrieval und metadaten-basierten Methoden zur Abfrage der Datenbankinformation im WWW ist in [Loe00b] beschrieben.

- *Interaktive Visualisierung*: Das Arbeiten mit 3D Dokumenten setzt Methoden zur 3D Visualisierung voraus, d.h. daß durch den *Rendering*-Prozeß aus der Geometrie 2D Ansichten mit einem Modell zur Beleuchtungssimulation erzeugt werden. Die *Rendering-Pipeline* ([FDFH90]) kann vollständig auf dem Server ausgeführt (*Remote*-Visualisierung) werden, so daß Bilddaten zum Client übertragen werden, oder beim Client lokal ablaufen (lokale 3D Visualisierung). In diesem Fall werden einmalig die Geometriedaten zum Client übertragen. Auch die Verteilung der *Rendering-Pipeline* auf mehrere Rechner ist möglich. Die Interaktion geschieht über Aktionsnachrichten, die mit Hilfe einer Visualisierungsanwendung auf die 3D Szene übertragen werden, und anschließend erneute Bilderzeugung. Durch diese interaktive *Rendering*-Schleife entsteht beim Benutzer der Eindruck einer kontinuierlichen Interaktion mit der 3D Szene.

- *Kommunikation*: Hier wird zwischen der Kommunikation zwischen Client und DL-Server zur Aushandlung von Rechten und lieferbaren Objektrepräsentationen und der Kommunikation zwischen mehreren Benutzern unterschieden. Der zweite Fall umfaßt Audio/Video-Konferenzen zwischen kooperierenden Benutzern und kann z.B. mit MBone-Tools unabhängig von der Visualisierungsanwendung realisiert werden. Dieser Aspekt soll hier jedoch nicht weiter untersucht werden. Die Kommunikation zwischen Client und Server wird jedoch einen wichtigen Teil der Architektur SCA3D darstellen und in den weiteren Kapitel genauer diskutiert.

Von den genannten Aspekten und Methoden zur Unterstützung verteilten Arbeitens mit digitalen 3D Dokumenten wird in der weiteren Arbeit vor allem der Punkt interaktive Visualisierung im Mittelpunkt stehen und untersucht werden. Retrieval mit Hilfe von metadaten- bzw. inhaltsbasierten Verfahren und Audio/Video-Kommunikation werden nur am Rande erwähnt, da diese Punkte nicht zentral für die wissenschaftliche Fragestellung der Arbeit sind.

Anforderungen für das Arbeiten mit digitalen Dokumenten in verteilten Informationsräumen

In Abbildung 2.8 ist ein verteiltes System einer Digitalen Bibliothek skizziert. Mit dem DL-System, das Dokumente und Dienste verwaltet und anbietet, sind Benutzer mit ihrem Computer über ein Telekommunikationsnetz verbunden. Ihr Ziel ist das interaktive Arbeiten mit den digitalen 3D Dokumenten innerhalb des skizzierten Informationsraumes. In einem realen DL-System sind die Aspekte Sicherheit, Bandbreite der Anbindung und Leistung der Client-Rechner im Hinblick auf Hardware und Software von großer Bedeutung. Diese kritischen Punkte sind in Abbildung 2.8 bei den jeweiligen Komponenten des Gesamtsystems eingezeichnet. Die Anforderungen an eine Software-Architektur, die sich aus diesen Aspekten ergeben, sollen nun diskutiert werden.

Die Sicherheitsanforderungen, die für die hier untersuchte Software-Architektur hervorgehoben werden sollen, sind der Urheberrechtsschutz (IPR: *Intellectual Property Rights protection*), die Zugriffskontrolle auf Dokumente am Server, die Integrität und die Vertraulichkeit von Daten in dem verteilten System einer Digitalen Bibliothek. Eine Diskussion dieser Aspekte wurde in Abschnitt

Abbildung 2.8: Skizzierung einer DL-Umgebung mit den kritischen Aspekten

2.4 vorgenommen und soll an späterer Stelle im Bezug auf die notwendigen Methoden in Kapitel 3 weitergeführt werden. Als Anforderung an eine Software-Architektur soll hier die Möglichkeit zur Berücksichtigung der genannten Sicherheitsaspekte festgehalten werden.

Da zur Übertragung von Dokumenten und zur Kommunikation ein Netzwerk benötigt wird, spielt die Anbindung bzw. die verfügbare Bandbreite zwischen Server und Client eine große Rolle für das Gesamtsystem. An gleicher Stelle kann der dritte Aspekt, nämlich die lokale Leistung, angesprochen werden, der sich auf die sehr unterschiedliche Leistung von Computern in heterogenen Umgebungen bezieht. Eine Anforderung an eine Software-Architektur, die das verteilte Arbeiten mit geschützten Dokumenten erlauben soll, muß also die Anpassungsfähigkeit des System an die Randbedingungen wie Bandbreite (evtl. garantierte Qualität) und lokale Leistung enthalten.

Eine solche Architektur, die mehrere Benutzer synchronisiert mit Informationen versorgen und sich an die Bedingungen eines einzelnen Clients anpassen soll, muß über eine Möglichkeit zur kontinuierlichen Auswertung der Güte eines Clients verfügen. Diese Güte wird hier als skalarer Wert mit Hilfe einer Performanzmetrik bestimmt und bezieht sowohl die Anbindung als auch die Verarbeitungsleistung eines Benutzerrechners mit ein. Als globales Kriterium, das optimiert werden soll, kann dabei die Verzögerung zwischen Server und Client während des Arbeitens dienen. Weiterhin müssen Parameter gefunden werden, die an die Randbedingungen der jeweiligen Situation angepaßt werden können, um so optimale Bedingungen zu schaffen. Diese Idee wird in der Software-Architektur der Skalierbaren Szenen umgesetzt, die im Kapitel 6 beschrieben wird.

Die Anforderungen an eine verteilte DL-Umgebung sollen hier noch einmal zusammengefaßt werden:

- *Verteilte Visualisierung geschützter 3D Dokumente*: Da die Dokumente zur 3D Visualisierung heute in vielen Fällen auf die lokalen Computer übertragen werden müssen und gleichzeitig Sicherheitsaspekte, wie Urheberrechte, berücksichtigt werden sollen, müssen Möglichkeiten zur Sicherung dieser Dokumente in eine verteilten Visualisierungs-Architektur integriert sein.

- *Dynamische Berücksichtigung von Anbindung und Leistung der Clienten*: Um die Anpassungsfähigkeit der Software-Architektur an die jeweiligen Randbedingungen eines Benutzers zu ermöglichen, muß eine Güte mittels einer Performanzmetrik ermittelt und im System bekannt gemacht werden. Die Architektur muß über Parameter verfügen, die für die jeweilige Situation optimiert werden können. Im Fall der Software-Architektur SCA3D werden diese Parameter mit verschiedenen Dokumentrepräsentationen und deren Auflösungsgrad verknüpft sein. Dieses Konzept wird hier als *Level-of-Information* bezeichnet und im weiteren genauer untersucht.

Kapitel 3

Methoden

In diesem Kapitel sollen die wichtigsten Methoden, die im Rahmen der Arbeit benötigt werden, näher beschrieben werden. Grundlegende Methoden der 3D Visualisierung in verteilten Umgebungen werden vorgestellt und Techniken zur Bildcodierung und Übertragung von codierten Bilddatenströmen erläutert. Weiterhin werden Methoden zum Entwurf von verteilten Software-Architekturen, soweit sie in der Arbeit benötigt werden, beschrieben.

3.1 Geometrische Repräsentationen von virtuellen 3D Objekten

Zunächst soll genauer auf verschiedene Repäsentationsformen von 3D Objekten in virtuelle Umgebungen eingegangen werden. Wie in Abschnitt 2.3 beschrieben, enthalten 3D Dokumente neben den übergeordneten Strukturinformationen (Szenengraph) und den Materialeigenschaften auch den eigentlichen geometrischen Inhalt, nämlich die verschiedenen Objekte einer Szene. In diesem Abschnitt sollen Möglichkeiten für die Beschreibung der geometrischen Parameter dieser 3D Objekte erläutert werden. Die wichtigsten und gebräuchlichsten Repräsentationen sind polygonale 3D Netze, parametrisierte Kurven bzw. Flächen und *Quadric Surfaces*, die mathematisch exakt beschreibbare Objekte darstellen. In 3D Szenen können jeweils verschiedene Beschreibungsformen parallel benutzt werden. Weitere Repräsentationen von 3D Objekten, die durch Kombinationen aus 3D und 2D Informationen unter Erhaltung wichtiger Objekteigenschaften entstehen, werden ebenfalls erläutert. Im folgenden werden diese Hauptbeschreibungsarten vorgestellt und ihre Vor- und Nachteile diskutiert. Ein aktueller Überblick der Verfahren und Repräsentationsformen von Oberflächen zur geometrischen Modellierung wird in Hubeli und Gross [HG00] gegeben.

3.1.1 Polygonale 3D Netze

Polygonale 3D Netze bestehen aus einem Satz von verbundenen Oberflächenelementen, die jeweils von Polygonen umfaßt werden und gemeinsame Kanten aufweisen. Die Polygone sind durch Eckpunkte (Vertex) im dreidimensionalen Raum gegeben und weisen eine bestimmte Umlaufrichtung auf. Jeweils zwei Eckpunkte sind entlang des Polygonzuges durch eine Kante (*Edge*) verbunden. Es existieren verschiedene Repräsentationsformen von polygonalen Netzen, von denen die drei wichtigsten hier genannt werden sollen. In Abbildung 3.1 sind sie anhand eines Dreiecksnetzes aus zwei Polygonen illustriert.

- *Explizite Beschreibung*: Hierbei werden für jedes Polygon die Eckpunkte (x, y, z) in Form einer Liste zusammengefaßt, wobei die Reihenfolge der Eckpunkte entlang der Kanten des

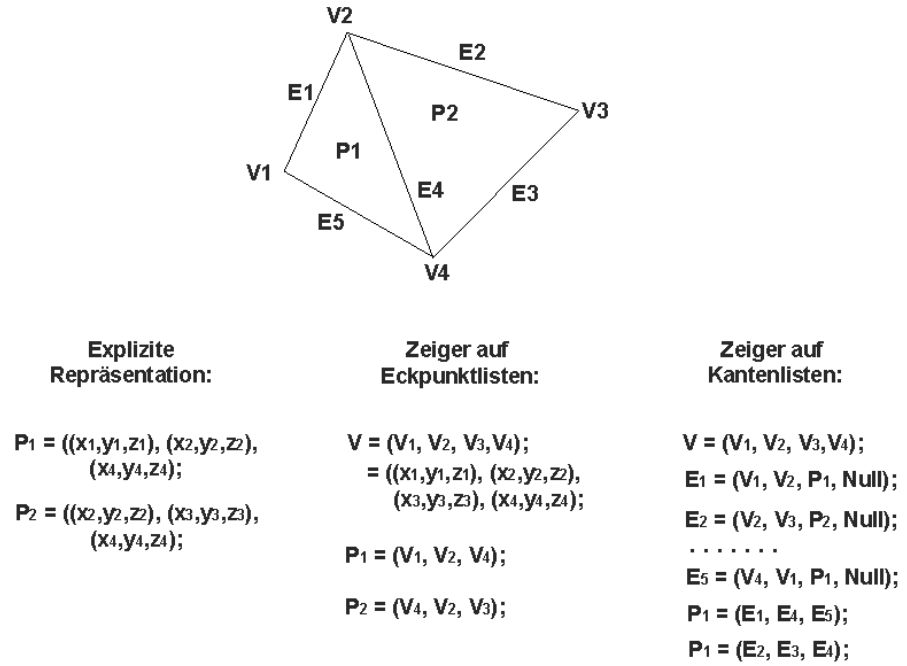

Abbildung 3.1: Beschreibungsformen für polygonale 3D Netze.

Polygons der Anordnung in der Liste entspricht. Zur Speicherung von großen Netzen ist dieses Verfahren sehr speicherintensiv, da Eckpunkte mehrfach gespeichert werden müssen. Es gibt auch keine direkte Information über gemeinsame Kanten und Eckpunkte von Polygonen, was dazu führen kann, daß beim Darstellen auf dem Bildschirm Kanten zweimal gezeichnet werden. Vorteile bietet die explizite Beschreibung bei der Konsistenzkontrolle von Netzen, da sie von den hier diskutierten Beschreibungen am meisten Information enthält. In Abbildung 3.1 ist auf der linken Seite ein Beispiel für explizite Angabe von Polygonen gezeigt.

- *Beschreibung durch Zeiger auf Eckpunktlisten*: Bei dieser Beschreibungsform werden alle Eckpunkte eines Netzes in einer gemeinsamen Liste V erfaßt, wie in Abbildung 3.1 in der Mitte gezeigt. Ein Polygon wird durch eine Liste von Indizes oder Zeigern auf die Eckpunktliste definiert. In Abbildung 3.1 ist P_1 durch die Eckpunkte V_1, V_2 und V_4 gegeben. Die Beschreibung durch Zeiger auf einen Eckpunktliste hat gegenüber der expliziten Form die Vorteile, daß jeder Eckpunkt nur einmal gespeichert werden muß und daß Koordinaten von Eckpunkten sehr einfach geändert werden können. Es ist aber weiterhin schwierig, Polygone mit gemeinsamen Kanten zu finden und gemeinsame Kanten werden unter Umständen auch noch doppelt gezeichnet.

- *Beschreibung durch Zeiger auf Kantenlisten*: Die Repräsentation von Polygonnetzen durch Zeiger auf Kantenlisten verwendet ebenfalls eine Eckpunktliste V. Darüber hinaus wird aber eine Kantenliste benutzt, die für jede Kante Angaben über die Eckpunkte und die zugehörigen Polygone der Kante enthält. Ein Polygon wird dann durch eine Liste von Zeigern auf seine

jeweiligen Kanten in der Kantenliste definiert. Dies ist in Abbildung 3.1 auf der rechten Seite dargestellt. Polygone werden dann durch die Zeichnung aller Kanten dargestellt, wodurch keine doppelt gezeichneten Kanten mehr auftreten.

Ein wichtiger Aspekt für polygonale Netze ist die Überprüfbarkeit der Konsistenz, also ob alle Polygone geschlossen sind, alle Eckpunkte mindestens einmal und nicht öfter als ein maximaler Wert verwendet werden und ob jeder Eckpunkt in mindestens zwei Kanten vorkommt. Weiterhin ist es sehr oft wichtig, daß das Netz komplett verbunden ist, daß es topologisch planar ist und keine Löcher aufweist. Diese Punkte müssen für ein Netz nach der Erzeugung durchgeführt werden und erfordern je nach Repräsenationsform einen unterschiedlichen Aufwand (siehe hierzu in [FDFH90] Abschnitt: *Consistency of Polygon-Mesh Representations*).

3.1.2 Parametrisierte Kurven und Flächen

Mit Polygonen, die stückweise lineare Approximationen erster Ordnung an Flächen darstellen, können nur ebenfalls stückweise lineare Flächen exakt dargestellt werden. Für alle anderen Flächen müssen sehr große Mengen an Stützpunkten erzeugt und gespeichert werden, um eine zufriedenstellende Genauigkeit der Approximation zu gewährleisten. Um ein in solcher Weise dargestelltes Objekt interaktiv verändern zu können, müssen sehr viele Punkte angepaßt werden, so daß die entsprechenden Operationen rechenintesiv werden. Eine Alternative ist die Verwendung von mathematischen Funktionen höherer Ordnung, mit denen Kurven und Flächen effizienter angenähert werden können und die sich daher besser zur Speicherung und Manipulation eignen. Dies soll nun anhand von Kurven im dreidimensionalen Raum erläutert werden; Flächen stellen eine Generalisierung des Ansatzes dar. Sehr häufig werden parametrisierte Repräsentationen von Kurven mit Polynomen dritter Ordnung für jede Komponente x, y und z in der Form verwendet:

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x, \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y, \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z, \qquad \text{mit} \qquad 0 \leq t \leq 1 \end{aligned} \tag{3.1}$$

Kubische Polynome werden verwendet, weil Funktionen mit geringerer Ordnung keine flexible Kontrolle der Form einer Kurve zulassen und Funktionen höherer Ordnung schnell sehr schwierig kontrollierbar werden. Parametrische Kurve ersetzten den Gebrauch der geometrischen Steigung von Funktionen, die auch unendlich werden kann, durch die Verwendung parametrisierter Tangentenvektoren, die besser handhabbar sind. Zur vollständigen Bestimmung einer kubischen parametrisierten Kurve sind vier bekannte Größen nötig. Dies können die beiden Endpunkte einer Kurve und die jeweiligen Tangentenvektoren an den Endpunkten sein, wie im Fall der Hermite-Kurven. Neben den Hermite-Kurven sind auch die Bézier-Kurven und die Splines sehr gebräuchlich, die allesamt parametrisierte Kurven unter Verwendung kubischer Polynome darstellen und sich in den Angaben der bekannten Randbedingungen unterscheiden. Werden Kurvensegmente $Q(t)$ aneinandergefügt, so existieren Bedingungen für die Randbedingungen, die entscheiden, in welcher Form die Kurvensegmente ineinanderübergehen. Die wichtigsten sind C^0 (ein gemeinsamer Endpunkt) , C^1 (gemeinsamer Endpunkt, Tangentenvektoren in diesem Punkt kollinear und mit gleicher Länge), C^2 (gemeinsamer Endpunkt, Tangentenvektoren in diesem Punkt kollinear und mit gleicher Länge, zweite Ableitung von Q(t) nach t am Übergang gleich) und G^1 Kontinuität ((gemeinsamer Endpunkt, Tangentenvektoren in diesem Punkt kollinear, aber mit ungleicher Länge)).

Für ein Kurvensegment $Q(t)$ kann man das Gleichungssystem 3.1 folgendermaßen in Matrizenschreibweise ausdrücken:

$$\begin{aligned} Q(t) &= T * C \\ Q(t) &= [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \end{aligned} \tag{3.2}$$

Die Koeffizientenmatrix C kann wiederum als Produkt einer Geometriematrix G und einer Basismatrix M dargestellt werden: $Q(t) = T * M * G$. Die Geometriematrix enthält dabei die vier geometrischen Randbedingungen einer Beschreibungsform.

- *Hermite-Kurven*: Ein Kurvensegment in der Hermite-Form wird durch die beiden Endpunkte P_1 und P_2 des Segmentes und durch die Tangentenvektoren T_1 und T_2 in den Endpunkten beschrieben. Durch Kontrolle der Endpunkte und Tangentenvektoren kann ein Anwender die Kurvenform kontrollieren.

- *Bézier-Kurven*: Bei der Bézier-Form werden neben den beiden Endpunkten P_1 und P_4 noch weitere Kontrollpunkte P_2 und P_3 angegeben, die nicht auf der Kurve selber liegen und die Tangentenvektoren P_1P_2 bzw. P_3P_4 in den Endpunkten festlegen.

- *Splines*: Splines leiten sich von der z.B. im Bootsbau verwendeten Technik ab, Metallstreifen bzw. Holzleisten durch Kontrollpunkte in eine glatte Kurvenform zu bringen. Ein Spline ist durch mehrere Kurvensegmente definiert, die durch jeweils vier Kontrollpunkte festgelegt werden und deren Übergänge C^2-Kontinuität aufweisen. Die Kontrollpunkte liegen dabei im allgemeinen nicht auf der Kurve. Mit uniformen, nicht-rationalen B-Splines (NURBS) können folgendermaßen kontinuierliche Kurven erzeugt werden. Der erste Kurvenabschnitt wird mit entsprechenden Kontrollpunkten festgelegt. Alle folgenden Kurvensegmente erhalten als erste Kontrollpunkte die letzten drei Kontrollpunkte des vorhergehenden Segmentes und der letzte Kontrollpunkt kann frei gewählt werden. Die Kontrollpunkte sind jeweils in der Geometriematrix G abgelegt. Weitere Details zu Splines sind in [FDFH90] beschrieben.

3.1.3 Unterteilung von parametrisierten Kurven und Flächen: *Subdivision Surfaces*

Die Modellierung von Kurven oder Oberflächen durch verbundene, parametrisierte Segmente mit einer festen Anzahl von Kontrollpunkten ermöglicht eine Annäherung mit bestimmter Genauigkeit. Wenn beim Modellieren eine bessere Approximation einer angestrebten Kurvenform bzw. Oberflächenform erzielt werden soll, so kann dies durch eine feinere Unterteilung des Netzes der Kontrollpunkte erreicht werden. Aus einem Bézier-Kurvensegment, das durch vier Kontrollpunkte festgelegt ist, werden nun z.B. zwei Segmente, die insgesamt durch sieben Kontrollpunkte bestimmt sind, da sie sich einen Kontrollpunkt teilen. Das selbe Prinzip wird auch für parametrisierte Flächen angewendet, um ein optimales Kontrollnetz zu erhalten. In diesem Fall spricht man von *Subdivision Surfaces.* Diese entstehen durch unendliche Verfeinerung der ursprünglichen Kontrollnetze durch Unterteilung der Maschen und durch Glättung. Aus einem Ausgangsnetz können durch endliche Schritte sehr gute Annäherungen an Freiformobjekte erzielt werden.

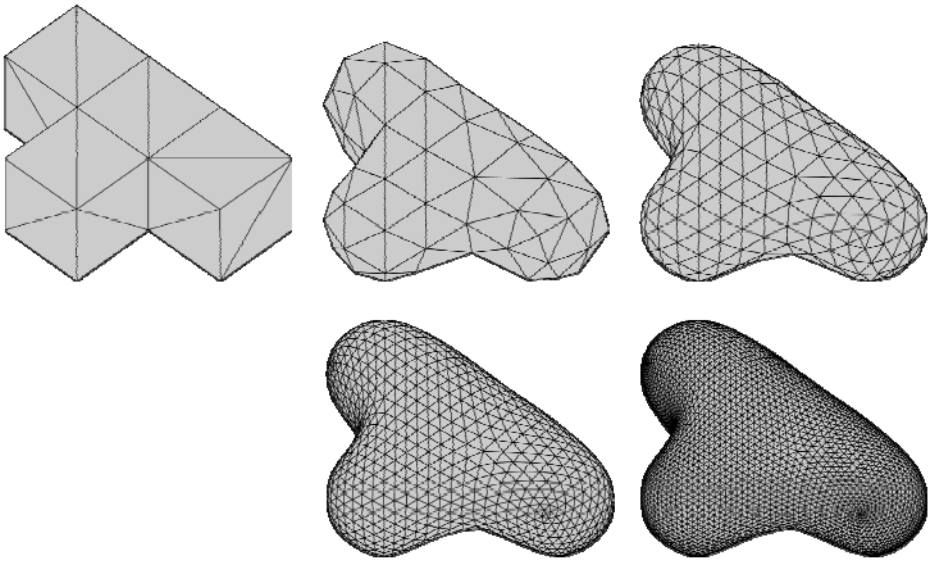

Abbildung 3.2: Modellierung eines Objektes durch Unterteilung nach dem *Loop*-Schema (Abbildung aus [Gro98]).

Die beiden bekanntesten Unterteilungsverfahren für Flächen stammen von Loop ([Loop87]) und Catmull-Clark ([CC78]). *Loop-Subdivision-Surfaces* arbeiten mit Dreiecksnetzen, während das Catmull-Clark-Verfahren Kontrollnetze mit rechteckigen Maschen verwendet. In Abbildung 3.2 ist die schrittweise Approximation eines Freiformobjektes aus einem Ausgangsnetz bestehend aus mehreren Würfeln gezeigt (Abbildung aus [Gro98]). Fast jedes polygonale Netz kann als Kontrollnetz verwendet werden, um ein Freiformobjekt anzunähern. Die Bedingungen, die an ein Netz für Loop-Unterteilungsflächen gestellt werden, sind in [MH00] beschrieben.

3.1.4 Implizite Funktionsdarstellung: *Quadric Surfaces*

Kurven und Flächen können als Lösung impliziter Gleichungen der Form

$$f(x, y, z) = 0 \tag{3.3}$$

modelliert werden. Ein einfaches Beispiel ist eine Kugel mit dem Radius 1, deren Oberfläche durch die Gleichung $x^2 + y^2 + z^2 - 1 = 0$ beschrieben wird. Die Vorteile dieser Darstellung liegen in ihrer guten Handhabbarkeit, z.B. bei der einfachen Berechnung der Oberflächennormalen durch $[\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}]$. Nachteile treten dadurch auf, daß es mehr als die gewünschten Lösungen für die gewählte Gleichung geben kann, wie z.B. bei der Modellierung einer Halbkugel, wobei bestimmte Randbedingungen eingeführt werden müssen, hier $z \geq 0$.

Die allgemeine implizite Oberflächengleichung der Form:

$$f(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0 \tag{3.4}$$

definiert die Familie der *Quadric Surfaces*. Durch Wahl geeigneter Koeffizienten können vielfältige Oberflächen modelliert werden, z.B. ergibt sich mit $a = b = c = -k = 1$ und ver-

schwindenden anderen Koeffizienten die obige Darstellung der Einheitskugel im Ursprung. Weitere Vorteile dieser Beschreibungsform ergeben sich durch Möglichkeiten für effiziente Tests, ob ein Punkt auf der Oberfläche liegt, zur Berechnung von z-Werten bei gegebenem x, y und zur Berechnung von Schnitten mit anderen Flächen.

3.1.5 Erweiterung von Objektrepräsentationen

Bei der Visualisierung von 3D Szenen werden die Prozesse der Visualisierungs-Pipeline, die sich je nach Visualisierungsverfahren unterscheiden kann (siehe [FDFH90]: Kapitel *Illumination and Shading*), schrittweise durchgegangen. Als Eingabe dient der Visualisierungs-Pipeline die 3D Szene, die durch die Szenengraphstruktur, die geometrischen Parameter der jeweiligen Objektrepräsentation und die Materialeigenschaften beschrieben ist. Die geometrischen Parameter können dabei implizit bzw. parametrisch angegeben werden oder explizit als 3D Netz planarer Polygone. Diese 3D Netze (z.B. Dreiecksnetze) können durch verschiedene Verfahren, wie 3D Scanning realer Objekte oder Tesselierung von parametrisierten Oberflächen, erzeugt worden sein. Nach den nötigen Visualisierungsschritten werden Bildobjekte bzw. Bilddaten ausgegeben, die eine Visualisierung der 3D Szene unter den Bedingungen des gewählten Beleuchtungsmodells und der Kameraparameter darstellen.

Bei der Visualisierung entstehen aber weitere Repräsentationsformen von Objekten, die zwischen der vollständigen 3D Beschreibung und der resultierenden 2D Beschreibung liegen und einen unterschiedlichen Gehalt an Information über die Objekte enthalten. Prinzipiell werden vom 3D Modell bis zum visualisierten Bild schrittweise immer mehr explizite Objekteigenschaften aus der Objektrepräsentation entfernt, so daß Objekte im Bild mit ihrer semantischen Bedeutung schließlich nur noch durch den menschlichen Wahrnehmungsapparat interpretiert werden können. Für den Benutzer, der mit einer 3D Szene umgeht, hat dies natürlich den Vorteil, daß die nicht direkt interpretierbaren geometrischen Beschreibungsformen von Objekten nun visuell verarbeitet werden können. Aus verschiedenen Gründen, die hier vorallem durch den abgestuften Informationsgehalt gegeben sind, ist es aber auch von Vorteil Objektrepräsentationen zur Verfügung zu stellen, die zwischen der analytischen bzw. numerischen Beschreibung und der visuellen Beschreibung angesiedelt sind und während der Visualisierung als Zwischenprodukte anfallen. Bei der 3D Visualisierung wird eine Merkmalsextraktion (*Feature Extraction*) für die Objekte einer Szene durchgeführt, die für verschiedene Zwecke genutzt werden kann. Als mögliche Zwischenrepräsentationen werden hier 3D Netze mit verschiedener Auflösung, Bounding-Box-Informationen, Konturen bzw. Masken und Texturen von Objekten angesehen. Diese können wiederum kombiniert werden, um neue Beschreibungsformen zu erhalten, z.B. Bounding-Box-Informationen plus Texturen. Das Ziel ist dabei die Erhaltung von Objekteigenschaften, während der explizite Informationsgehalt der Beschreibung schrittweise abnimmt (*Level-of-Information*).

In Abbildung 3.3 ist dieses Konzept anhand von zwei Visualisierungs-Pipelines dargestellt. Als Eingabe erhält die erste Pipeline ein 3D Objekt mit geometrischen Parametern und Materialeigenschaften. Am Ende der Pipeline werden Bilddaten auf dem Bildschirm oder als Daten zur Weiterverarbeitung ausgegeben. Während der Bearbeitung in der Visualisierungs-Kette entstehen zusätzliche Informationen über die Objekte, die bei Bedarf separat weiterverarbeitet werden können. In der Abbildung sind dies 3D Netze, Bounding-Box-Daten, Bitmasken und schließlich Texturen. Die Objekteigenschaften werden bei dieser Abfolge teilweise erhalten, so daß sie durch Integration in eine zweite Visualisierungs-Pipeline als Objekt visualisiert werden können. Der rechnerische Aufwand der Visualisierung kann auf diese Weise in der zweiten Pipeline an lokale Bedingungen angepaßt werden. Weiterhin können Informationen, die geschützt sind, durch eine veränderte Beschreibungs-

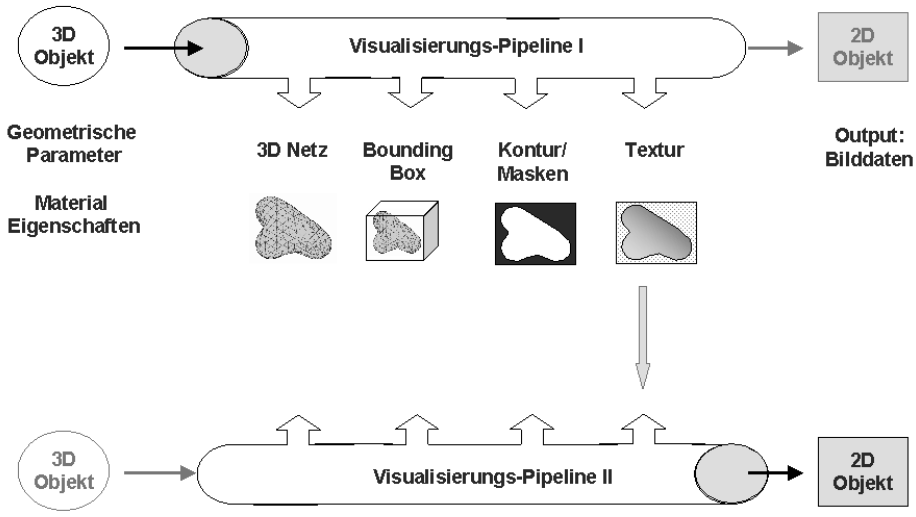

Abbildung 3.3: Erweiterte Repäsentationen zur Erhaltung von Objekteigenschaften zwischen zwei Visualisierungs-Pipelines.

form von Objekten flexibel verwaltet werden.

Für das verteilte Arbeiten mit 3D Dokumenten bietet dieses Konzept aber noch weitere Vorteile, da z.B. für das Retrieval von 3D Objekten in verteilten Umgebungen Objekteigenschaften benötigt werden, die sich in Form von standardisierbarer Merkmalen in festgelegten Schemata ausdrücken lassen. In [PR99] werden verschiedene Merkmale, wie Volumen, Bounding-Box, Bounding-Sphere und der Trägheitstensor bezüglich einer geometrischen Form, zum Auffinden von 3D Objekten in verteilten Informationsräumen diskutiert, die in den entstehenden MPEG-7 Standard ([ISO00a]) eingehen sollen. In [Loe00b] werden vom Autor der vorliegeden Arbeit Konturinformationen von 3D Objekten verwendet, um einen visuellen Index für eine 3D Datenbank zu erzeugen, der von entfernten Benutzer zum Retrieval verwendet wird.

3.2 Digitale Bildsynthese und interaktive 3D Visualisierung

Interaktive 3D Visualisierung besteht im wesentlich aus zwei Prozeßabfolgen, die in gegengesetzter Richtung zwischen der 3D Szene und der Benutzerschnittstelle ablaufen. Von der Eingabe der 3D Szene zur Benutzerschnittstelle, z.B. einem nicht-immersiven Display oder einer immersiven *Responsive Workbench* [GFG98], werden entlang der Rendering-Pipeline Bilder erzeugt, die Ergebnisse der verwendeten Beleuchtungssimulation darstellen. Diese Bilder werden beim Benutzer in der gewählten Form angezeigt. Es können nun Aktionen durch definierte Eingaben bezüglich des visuell wahrgenommenen 3D Modells erzeugt werden, die wiederum über eine Prozeßabfolge, hier als Aktions-Pipeline bezeichnet, auf die eigentliche 3D Szene übertragen werden. Durch erneute Visualisierung erhält der Benutzer ein visuelles Feedback bezüglich seiner Aktionen, das er zum interaktiven Arbeiten mit der 3D Szene benötigt. Die beiden Richtungen der Prozeßabläufe sind in

Abbildung 3.4 skizziert.

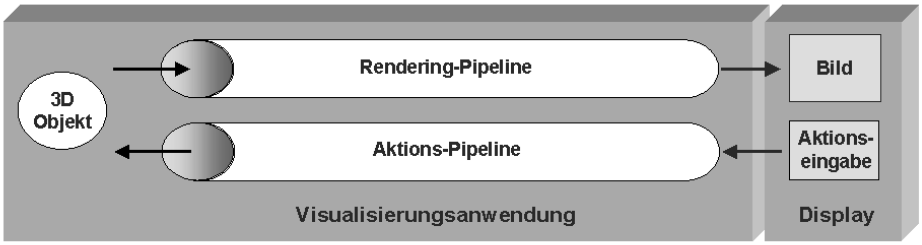

Abbildung 3.4: Visualisierungs- und Aktions-Pipeline bei der interaktiven Visualisierung.

Eine allgemeine Modellierung der Datenvisualisierung wird von Haber und McNabb [HM90] angegeben, die folgende drei Hauptkomponenten enthält: *Data Enrichement, Visualization Mapping* und *Rendering.* Eine neuere Terminologie in [BDG98] benennt diese drei Prozesse als Datenfilterung (*Filter*), Datenabbildung (*Map*) und Darstellung (*Render*). Filterung meint dabei, die Extrahierung und Aufbereitung von Rohdaten, z.B. Auswahl von allen Punkten einer 3D Punktwolke, die zu einem bestimmten Objekt gehören, oder Ermittlung einer Iso-Fläche für Volumendaten. Der nächste Schritt, die Datenabbildung, erzeugt aus den angereicherten Daten eine sogenanntes abstraktes visuelles Objekt AVO (*Abstract Visual Object*), das aus Attributen, wie z.B. Geometrieelementen und Materialeigenschaften, besteht, die aus den quantitativen Eigenschaften der Daten abgeleitet werden. Nach diesem Schritt wird das AVO durch einen Darstellungsprozeß in eine vom Benutzer visuell interpretierbare Form gebracht, also z.B. durch eine 3D Rendering-Pipeline als Bild auf einem Display ausgegeben. Diese drei wichtigen Schritte sind in Abbildung 3.5 gezeigt.

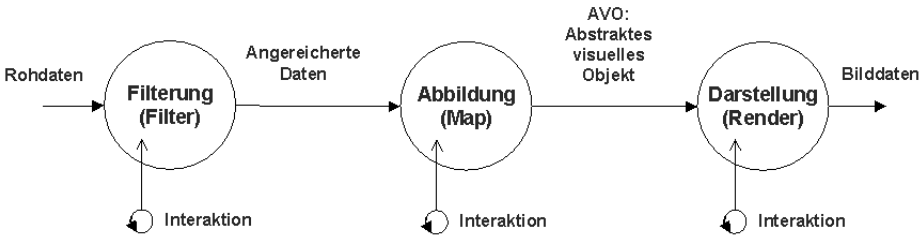

Abbildung 3.5: Allgemeines Visualisierungsmodell nach Haber und McNabb [HM90].

Die Interaktion des Benutzer mit den Daten kann nun an verschiedenen Stellen des in Abbildung 3.5 angegebenen Modells stattfinden. Larkin et al. [LHG97] erläutern, daß Interaktion durch die dynamische Kontrolle der Prozesse Filter, Map und Render des obigen Modells ermöglicht wird. Die vom Benutzer kontrollierten Parameter werden über die Aktions-Pipeline (vgl. Abbildung 3.4) an die jeweiligen Prozesse übergeben und damit auch auf die Datenrepräsentationen angewendet.

In Abbildung 3.5 ist dies durch Pfeile mit der Kennzeichnung *Interaktion* dargestellt. Ein Überblick über interaktive geometrie-basierte Visualisierungskonzepte anhand der wissenschaftlichen Literatur wird in Kapitel 4 gegeben.

Ein kritischer Faktor bei der interaktiven Visualisierung ist die Verzögerung, die durch die benötigte Zeitspanne zur Erzeugung und Darstellung neuer Bilddaten bestimmt wird. Dieser Faktor wird hier Visualisierungsverzögerung genannt. Die Verzögerung durch die Erzeugung, Übertragung und Anwendung der Aktionen auf das 3D Modell ist dagegen gering, da keine großen Datenmengen berechnet oder übertragen werden müssen. Dies gilt jedoch nur, solange keine komplizierten Modellierschritte am 3D Modell vorgenommen werden, die wiederum zeitintensiv sein können. Auch die Filter- und Map-Schritte aus Abbildung 3.5 können im Falle der Interaktion zu einer erhöhten Visualisierungsverzögerung beitragen.

Die Visualisierungsverzögerung wird stark durch das verwendete Beleuchtungsmodell und durch die Bildauflösung bestimmt. Natürlich steigt die Verzögerung stets mit der Komplexität der 3D Szenen an, durch Verwendung verschiedener Beleuchtungsmodelle werden aber entscheidende Unterschiede bei der Verzögerung in ein System eingeführt. Relativ einfache, lokale Beleuchtungsmodelle wie das Phong-Shading [Pho75] sind zur interaktiven Visualisierung gut geeignet, weil sie eine minimale Verzögerung bei ausreichender Bildqualität ermöglichen. Komplexere globale Beleuchtungsmodelle, wie das rekursive Raytracing oder die Radiosity-Methode, bieten eine realistische Darstellungsqualität, sind aber aufgrund des hohen benötigten Zeitaufwandes für die Berechnung eines Bildes für die interaktive Visualisierung nur beschränkt einsetzbar. Die Radiosity-Methode erlaubt allerdings interaktive *Walkthroughs* bei hoher Qualität, da die rechenintensiven Schritte im voraus durchgeführt werden können und eine Darstellung der beleuchteten Szene erzeugt wird, die unabhängig von Blickposition- und winkel ist.

3.2.1 Die Rendering-Pipeline

Der Prozeß der Bildsynthese für ein gegebenes digitales 3D Modell wird durch die Rendering-Pipeline beschrieben. Eine wichtige Komponente der Rendering-Pipeline ist die Beleuchtungssimulation, jedoch müssen weitere Schritte zur Visualisierung einer virtuellen 3D Szene durchgeführt werden. In Abhängigkeit vom gewählten Rendering-Algorithmus kann die Rendering-Pipeline unterschiedlich aufgebaut sein. Ein Vergleich hierzu ist in [FDFH90] zu finden. In Abbildung 3.6 ist ein Beispiel einer Rendering-Pipeline gezeigt, für das im folgenden eine allgemeine Diskussion der Einzelkomponenten durchgeführt wird.

Als Eingabe dient der Rendering-Pipeline eine 3D Szene, die aus 3D Objekten mit ihren geometrischen Parametern und Materialeigenschaften, Lichtquellen und mindestens einer Kameraeinstellung besteht. Diese Eingabeszene kann durch ein Modellierwerkzeug oder durch 3D Scanning entstanden sein und liegt in Form einer Szenenbeschreibung bzw. einer Szenendatenbank vor.

Beim **Szenendurchgang** (*scene traversal*) wird der strukturelle Aufbau der Szene erfaßt, um Transformationen der Objekte, wie Translation, Rotation und Skalierung, aufzulösen. Dabei werden die Objekte aus ihrem lokalen Koordinatensystem, in dem sie modelliert wurden, in das globale Weltkoordinatensystem transformiert, in dem die gesamte Beleuchtungsberechnung durchgeführt wird. Im nächsten Schritt wird mit Hilfe der Kameraeinstellung eine einfache Einteilung der Szene in relevante Objekte und nicht-relevante Objekte, die hinter dem Beobachter liegen, vorgenommen, um die Szenenkomplexität zu vermindern (**Annahme/Ablehnung**). Für lokale Beleuchtungsmodelle ist dieser Vorgang problemlos möglich, für eine globale Beleuchtungssimulation mit dem Raytracing- oder Radiosity-Verfahren müssen eventuell auch Mehrfachreflexionen berücksichtigt werden, die Einflüsse aus dem Bereich hinter dem Beobachter im Ergebnis sichtbar machen. Im An-

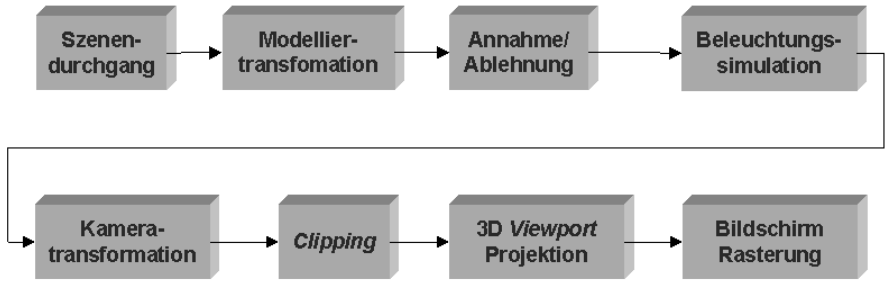

Abbildung 3.6: Rendering-Pipeline.

schluß an diesen Schritt folgt die **Beleuchtungssimulation**, welche eine möglichst gute Berechnung der Lichtausbreitung in dem beschriebenen geometrischen Raum liefern soll. Durch Verwendung von Raytracing- bzw. Radiosity-Verfahren können sehr realistische Darstellungen der Beleuchtung erreicht werden, jedoch steigt auch der Aufwand mit der erreichten Qualität stark an. In den unten folgenden Abschnitten werden lokale und globale Beleuchtungsmodelle näher besprochen.

Bei der folgenden **Kameratransformation** wird die gesamte 3D Szene in das Koordinatensystem der Kamera abgebildet. Beim sogenannten **Clipping** werden nun alle Eckpunkte des Modells entfernt, die nicht im Blickfeld (*viewing frustrum*) liegen. Das Blickfeld ist durch die Kameraposition, die Blickrichtung und die Öffnungswinkel bestimmt. Durch Projektion der verbleibenden Koordinatenpunkte auf das Fenster des Anzeigegerätes bzw. den *Viewport* der Anwendung (**3D Viewport Projektion**) werden die Berechnungen im dreidimensionalen Raum abgeschlossen. Der letzte Schritt der **Bildschirmrasterung** bestimmt für jeden Pixel des Gerätedisplays das entsprechende geometrische Primitiv aus den vorhergehenden Transformationen und ermittelt den zugehörigen Farbwert. Somit ensteht in diesem Schritt das eigentliche Bild auf dem Bildschirm. Die Farbwerte für jeden Bildpunkt wurden entweder mit der Beleuchtungssimulation bestimmt oder werden aus einem Texturbild entnommen (*texture mapping*).

Die genannten Schritte der Rendering-Pipeline werden entweder mit Hardware- oder Software-Komponenten durchgeführt. Hochleistungsrechner für Graphikanwendungen bieten oft eine komplette Hardware-unterstützte Rendering-Pipeline. Jedoch wird durch die schnelle Entwicklung der Rechenleistung von modernen Mikroprozessoren eine vollständige Ausführung durch Software-Komponenten für breite Anwendungsfelder immer realistischer.

3.2.2 Lokale Beleuchtungssimulation

Bei einem lokalem Beleuchtungsmodell hängen die berechneten Intensitäten an einem Oberflächenpunkt nur von den lokalen Oberflächen- und Materialeigenschaften und den beleuchtenden Lichtquellen ab. Um die Beleuchtung einer 3D Szene zu simulieren wird die gesamte Lichtintensität berechnet, die in die Richtung des Beobachters reflektiert wird. Die Gesamtintensität setzt sich aus einem diffusen, einem spekularen und einem ambienten Anteil zusammen, d.h. $I = I_d + I_s + I_a$. Die Reflexion von Licht an Oberflächen kann physikalisch durch diese spiegelnden (spekular) und diffusen Anteile beschrieben werden. Der diffuse Anteil hängt nur von der Intensität und Rich-

tung des Lichteinfalls ab, während er von der Beobachtungsrichtung unabhängig ist. Dies wird im Gesetz von Lambert ausgedrückt. Es besagt, daß der Betrag des reflektierten Lichtintensität proportional zum Cosinus des Winkels φ zwischen Oberflächennormalen N und Lichteinfallsrichtung L ist. In Abbildung 3.7 wird der Zusammenhang verdeutlicht. Für Winkel größer 90 Grad wird die Intensität theoretisch negativ. Da dies aber physikalisch nicht sinnvoll ist, werden nur Winkel zwischen 0 und 90 Grad einbezogen. Für alle anderen Fälle insbesondere auch um indirekte Beleuchtung annäherungsweise zu simulieren, wird ein konstanter Term eingeführt, der in einer ambienten Grundintensität resultiert.

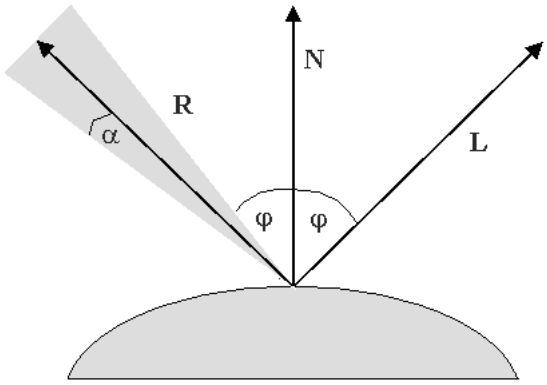

Abbildung 3.7: Lokale Beleuchtungssimulation mit spekularem Anteil.

Um spiegelnde Reflexion, also Spiegelungen und *Highlights*, simulieren zu können wird der spekulare Anteil in der Gesamtintensität mit Hilfe der Phong-Approximation [Pho75] berechnet. Spiegelnde Reflexion hängt im Gegensatz zur diffusen Reflexion von der Blickrichtung des Beobachters ab. Direkte Spiegelung und *Highlights* können innerhalb des Winkelbereichs eines Konus um die Hauptreflexionsrichtung R beobachtet werden (siehe Abbildung 3.7). Je nach Material ist dieser Konus enger oder weiter. In Hauptreflexionsrichtung ergibt sich die maximale Intensität der Reflexion, während nach Phong die Intensität mit der n-ten Potenz des Cosinus des Öffnungswinkels α des Konus abnimmt. Der Potenzwert n unterscheidet die verschiedenen spekularen Reflexionseigenschaften der verwendeten virtuellen Materialien.

Die vollständige Gleichung zur Berechnung der Gesamtintensität kann nun folgendermaßen formuliert werden, indem die diffusen, spekularen und ambienten Teile verwendet werden:

$$I_{total} = k_a I_{ambient} + \sum_{i=1}^{n} I_i (k_d \cos \varphi_i + k_s cos^n \alpha_i) \tag{3.5}$$

In Formel 3.5 drücken die Koeffizienten k_a, k_s und k_d jeweils die ambienten, diffusen und spekularen Reflexionseigenschaften der Materialien aus. Die Intensitäten I_i und $I_{ambient}$ stehen für die einfallenden Intensitäten aller relevanten Lichtquellen bzw. die ambiente Lichtintensität, wobei die

letztere hier nur eine ungenaue Approximation ist. Diese Gleichung zur lokalen Beleuchtungssimulation eignet sich auch für den Einsatz in interaktiven Visualisierungsanwendungen, da nur relativ wenige Berechnungsschritte pro geometrischem Primitiv ausgeführt werden müssen. Nachteile ergeben sich durch die mangelnde Realitätsnähe der erzeugten Bilder der 3D Szene, insbesondere Schatten, Mehrfachreflexionen und durchscheinende Materialien können nicht oder nur als zusätzlicher Effekt dargestellt werden.

3.2.3 Globale Beleuchtungssimulation

Um eine größere Realitätsnähe der erzeugten Bilder einer beleuchteten 3D Szene zu ermöglichen, müssen Verfahren zur Berücksichtigung von Mehrfachreflexionen zwischen allen Objekten einer Szene verwendet werden. Dies führt zu einem starken Anwachsen des Berechnungsaufwandes, läßt aber auch eine Simulation vieler physikalischer Prinzipien der Lichtausbreitung zu. Die wichtigsten Verfahren zur globalen Beleuchtungssimulation sind das rekursive Ray-Tracing, bei dem Strahlen durch die Szene verfolgt werden, und das Radiosity-Verfahren, das ein Gleichungssystem zur Berechnung des Energieflusses zwischen diffus reflektierenden Oberflächenelementen unter Annahme der Energieerhaltung löst, um die Beleuchtungsverhältnisse zu ermitteln. Beide Verfahren liefern sehr realistische Ergebnisse, wobei teilweise ein sehr hoher Zeitaufwand und bestimmte Beschränkungen hinsichtlich der Darstellung für die beiden Verfahren heute noch unumgänglich sind.

Ray-Tracing

Die globale Beleuchtungssimulation mittels Strahlverfolgung (*Ray Tracing*) wurde erstmals von Whitted [Whi98] realisiert und beschrieben. Beim rekursiven Ray-Tracing wird der fotographische Prozeß umgekehrt, der in der Strahlenoptik durch eine Ausbreitung von Lichtstrahlen von der Quelle über Mehrfachreflektionen zum Beobachterauge, dem Sensor, beschrieben wird. Beim Ray-Tracing werden Strahlen vom Blickpunkt durch die Bildschirmebene in die 3D Szene geschossen und ihr Weg dort durch Mehrfachreflexion weiterverfolgt.

Ein Einzelstrahl, der vom Beobachtungspunkt durch einen Bildschirmpixel in die Szene verfolgt wird, wird auf seinen ersten Schnittpunkt mit einem Objekt der 3D Szene hin untersucht. Trifft er kein Objekt, so wird der Farbwert des Pixels auf die Hintergrundfarbe gesetzt. Wenn jedoch ein Schnittpunkt gefunden wird, so werden von diesem Oberflächenpunkt des Objektes weitere Strahlen in die Szene geschickt und weiterverfolgt. Am Schnittpunkt werden alle zurückgelieferten Intensitäten der Nachfolgestrahlen aufsummiert, um die Gesamtintensität und damit den Farbwert für den Bildpunkt zu berechnen. Durch unterschiedliche Strahltypen können bestimmte Effekte der Lichtausbreitung simuliert werden. Durch Verfolgung von Strahlen direkt zu den Lichtquellen der Szene, kann der direkte Einfluß der Lichtquellen berücksichtigt werden. Dieser wird dann wie im vorigen Abschnitt besprochen nach dem Lambert'schen Gesetz errechnet. Durch Untersuchung dieser Strahlen auf Schnittpunkte mit anderen Objekten, können Schatteneffekte berücksichtigt werden (*shadow rays*). Auch spekulare Effekte können auf diese Weise ermittelt werden.

Um die diffuse Reflexion zu berechnen, werden vom ersten Schnittpunkt in alle Richtungen innerhalb der oberen Hemisphäre Nachfolgestrahlen verteilt. Es existieren verschiedene Ansätze zu Verteilung der Strahlen, die z.B. die Monte-Carlo-Verteilung verwenden. Die Nachfolgestrahlen werden dann auf neue Schnittpunkte mit anderen Objekten untersucht, um an diesen Schnittpunkten den oben beschriebenen Aufsummierungsprozeß der Lichteffekte durchzuführen. Durch Rückmeldung des Intensitätswertes an die jeweils letzte Rekursionsstufe wird die Gesamtintensität am ersten Objektschnittpunkt sehr genau ermittelt. In Abbildung 3.8 ist dieser Prozeß des rekursiven

Ray-Tracing anhand einer Szene mit zwei Objekten und einer Lichtquelle dargestellt.

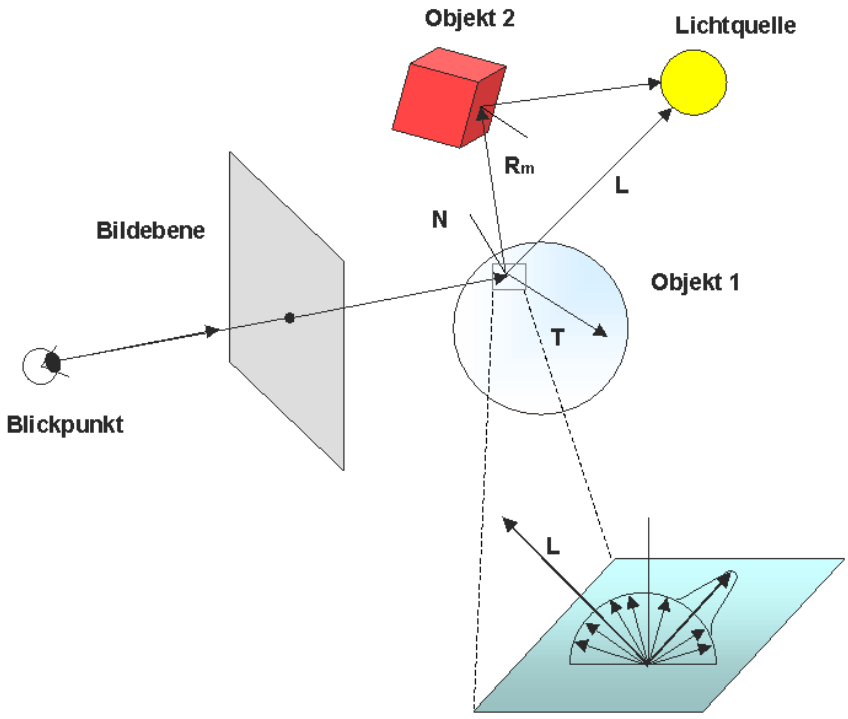

Abbildung 3.8: Globale Beleuchtungssimulation durch rekursives Ray-Tracing.

Objekt 1 ist dabei teilweise transparent und läßt eine Transmission T von einfallenden Lichtstrahlen zu. In der Vergrößerung des Schnittpunktes im unteren Abbildungsbereich ist die Strahlverteilung in die oberhalb des Oberflächenpunktes liegende Hemisphäre verdeutlicht. Die diffusen Anteile werden durch eine gleichmäßige Verteilung von Strahlen R_m simuliert, während der spiegelnd reflektierende Anteil durch die Phong-Approximation ermittelt wird.

Mit Hilfe des physikalischen Gesetzes von Snellius kann an Schnittpunkten nun auch die optische Brechung von Lichtstrahlen an teilweise transparenten Objekten berechnet werden. Das Gesetz von Snellius sagt aus, daß Lichtstrahlen beim Übergang von einem Medium mit Dichte δ_1 in ein Medium mit Dichte δ_2 teilweise reflektiert werden und sich teilweise durch das zweite Medium weiterausbreiten (Refraktion). Die reflektierte bzw. refraktierte Intensität hängt von dem Verhältnis der Brechungsindizes n_1 und n_2 ab, und weiterhin vom Einfallswinkel des Lichtstrahls. Wenn φ_1 der Winkel zwischen Einfallsrichtung und Oberflächennormale in Medium ist, und φ_2 der Winkel zwischen Transmissionrichtung und Oberflächennormale in Medium zwei ist, dann gilt nach Snellius:

$$n_1 \sin \varphi_1 = n_2 \sin \varphi_2 \tag{3.6}$$

Der rekursive Prozesse, der zur Summierung aller Effekte bei der Lichtausbreitung führt, kann auf verschiedene Weise abgeschlossen werden. Entweder wenn eine bestimmte Stufe der Rekursion

erreicht ist, kein Objekt mehr geschnitten wird oder der Beitrag eines Objektes unter eine kritische Schwelle fällt, wird die Rekursion abgebrochen und die Gesamtintensität für den entsprechenden Bildpunkt aufsummiert. Gleichung 3.7 gibt die Beleuchtungsformel für das Raytracing-Verfahren an, wobei die reflektierten und transmittierten Anteile der Intensität durch $k_s I_s$ bzw. $k_t I_t$ beschrieben werden. Für eine weitergehenden Diskussion siehe [Gla99].

$$I_{total} = k_a I_{ambient} + k_d \sum_{i=1}^{n} I_i \cos \varphi_i + k_s I_s + k_t I_t \tag{3.7}$$

Der rechnerische Hauptaufwand beim Ray-Tracing liegt in der Berechnung der Schnittpunkte zwischen Lichtstrahlen und Objekten. Daher wurde Verfahren zur optimalen Szenenstrukturierung und möglichst effizienten Schnittpunktberechnung entwickelt, die zuerst einfache Oberflächenformen von Würfeln oder Kugeln auf Schnittpunkte testen, ehe die darin eingeschlossenen Objekte der Szene untersucht werden. Dadurch können die Berechnungszeiten für ein Bild stark reduziert werden (siehe hierzu [MF99]).

Das Ray-Tracing-Verfahren erzeugt Darstellungen der beleuchteten Szene mit bestimmten Charakteristiken. Die Schattenbildung in der Szene ist sehr hart, da eine Lichtquelle von einem bestimmten Oberflächenpunkt aus voll sichtbar ist und für den Nachbarpunkt schon ganz verdeckt sein kann durch ein anderes Objekt. Weiche Übergänge von hell nach dunkel sind nur schwer zu erreichen. Durch die Verwendung des *Phong-Shading* erhalten die Oberflächen in den simulierten Bildern oft ein Erscheinungsbild, das an Plastikobjekte erinnert.

Die Radiosity-Methode

Der Begriff *Radiosity* bezeichnet bei diesem Verfahren den Strahlungsenergiefluß pro Flächeneinheit $\frac{d\phi}{dA}$ einer Objektoberfläche. Die globale Beleuchtung wird durch die Ermittelung des Strahlungsenergieflusses im Gleichgewicht zwischen allen diffusen Oberflächen einer 3D Szene berechnet. Durch diese Methode werden realistische Darstellungen der Beleuchtungssituation in Szenen erzielt, wobei im Gegensatz zum Ray-Tracing weiche Schattenübergänge ermöglicht werden. Das Radiosity-Verfahren wurde aus physikalischen Simulationsmethoden für den Strahlungsenergiefluß in geschlossenen Systemen für computergraphische Zwecke weiterentwickelt. Die ersten Algorithmen zur globalen Beleuchtungssimulation wurden von Goral et. al. [GTGB84] und Nishita und Nakamae [NN85] vorgestellt.

Da ein Berechnung des Energieflusses zwischen allen Oberflächenpunkten zu aufwendig wäre, werden die Objekte mit einem Netz von planaren Oberflächenelementen (*patches*) überzogen. Diese Oberflächenelemente stellen Lambert'sche Reflektoren dar, so daß nur diffuse Reflexion betrachtet wird. Lichtquellen werden als emittierende Oberflächenelemente dargestellt. Durch diese Vereinfachung läßt sich mit Hilfe der Radiosity-Methode eine Darstellung der beleuchteten 3D Szenen im Objektraum erzeugen, die unabhängig vom Blickpunkt ist. Mit dieser Methode lassen sich also interaktive *Walkthroughs* realisieren, da der rechenintensive Beleuchtungsschritt im voraus durchgeführt werden kann. Bei der interaktiven Visualisierung wird die Rendering-Pipeline dann unter Verwendung dieser Repräsentation der beleuchteten 3D Szene durchlaufen. Die Radiosity-Methode stellt, wie das Ray-Tracing, eine sehr zeitaufwendige Lösung für die globale Beleuchtungssimulation dar. Die rechnerisch Hauptarbeit liegt in der Ermittelung der Formfaktoren für eine gegebene Szene. Eine weiterführende Diskussion hierarchischer Radiosity-Methoden ist bei Schäfer [Sch00] zu finden

3.3 Verteilte, objekt-orientierte Systeme

In diesem Abschnitt werden Grundlagen objekt-orientierter, verteilter Systeme besprochen, die zum Entwurf einer Software-Architektur im Bereich Digitaler Bibliotheken benötigt werden. Verteilte Systeme im Umfeld der Informationsverarbeitung bestehen aus mehreren eigenständigen Rechnern, die über ein Kommunikationsnetzwerk miteinander kooperieren, um ein festgelegtes gemeinsames Ziel zu erreichen. Die Umsetzung verteilter Systeme erfolgt am besten in einer objekt-orientierten Sprache, die die Vorteile der Kapselung und wohl definierter Methodenschnittstellen bietet. Im folgenden werden zuerst die Eigenschaften solcher Systeme vorgestellt, dann folgt die Beschreibung des objekt-orientierten Modells und abschließend werden Verteilungsplattformen, sogenannte *Middleware*, am Beispiel des Industriestandards CORBA [OMG98] diskutiert.

3.3.1 Eigenschaften verteilter Systeme

Ein verteiltes System besteht aus einer Menge von verteilten, kooperierenden Komponenten. Die Kooperation der verschiedenen Komponenten wird über den Austausch von Nachrichten mittels eines Kommunikationsnetzwerkes realisiert. Diese Nachrichten werden zur Synchronisation der verteilten Prozesse und zum Datenaustausch verwendet. Neben den Prozessen, die auf verschiedenen Rechnern arbeiten, können auch die Daten einer Anwendung verteilt sein, indem sie z.B. aufgeteilt auf verschiedene Datenbanken im Gesamtsystem vorliegen. Schließlich können auch mehrere Benutzer gleichzeitig mit einem verteilten System arbeiten, indem sie Teile des Gesamtsystems auf ihrem Rechner betreiben. In diesem Fall spricht man von verteilten Benutzern.

Man unterscheidet die physikalische Sicht auf ein verteiltes System, die dessen technische Realisierung mit den beteiligten Rechnern als Knoten eines Kommunikationsnetzwerkes und die Eigenschaften des Netzwerkes selbst beschreibt. Im Gegensatz dazu beschreibt die logische Sicht die Anwendungsaspekte, die durch die Verteilung von Zustand (Daten) und Verhalten (Programmcode) bestimmt werden (siehe [CDK94]). In den Prozessen sind Teile des Zustandes und des Verhaltens gekapselt. In Abbildung 3.9 ist die Struktur eines verteilten Systems aus vier Rechnern und einem verbindenden Netzwerks gezeigt.

Verteilte Systeme weisen wichtige Vorteile aber auch Nachteile gegenüber zentral organisierten Systemen auf. Die Vorteile reichen von größerer Wirtschaftlichkeit durch ein besseres Preis-Leistungs-Verhältnis von kleineren, vernetzten Rechnern, über eine erhöhte Verfügbarkeit durch mögliche Redundanz, bessere Skalierbarkeit durch Erweiterbarkeit, bis hin zu Geschwindigkeitsvorteilen durch Parallelisierung von Programmteilen. Mögliche Nachteile entstehen durch die Offenheit, Komplexität und Heterogenität von verteilten Systemen, denn hierdurch nimmt die Anzahl potentieller Fehlerquellen zu, die die Funktionsfähigkeit des Gesamtsystems beeinträchtigen kann. Außerdem werden Angriffspunkte in einem solchen System leichter auszumachen und zu erreichen sein, so daß Sicherheitsaspekte eine besondere Bedeutung bekommen. Die Heterogenität erfordert in einem hohen Maße Mechanismen zur Integration verschiedener Technologien in einem verteilten System, z.B. können verschiedene Betriebssysteme und Programmiersprachen in verteilten Systemen gleichzeitig zum Einsatz kommen. Weiterhin kann die Kommunikation durch eine schlechte Netzverbindung verlangsamt werden, so daß eng zusammenarbeitende Komponenten möglichst nicht verteilt werden sollten. In Tabelle 3.1 sind die Vor- und Nachteile noch einmal in einer Übersicht zusammengefaßt.

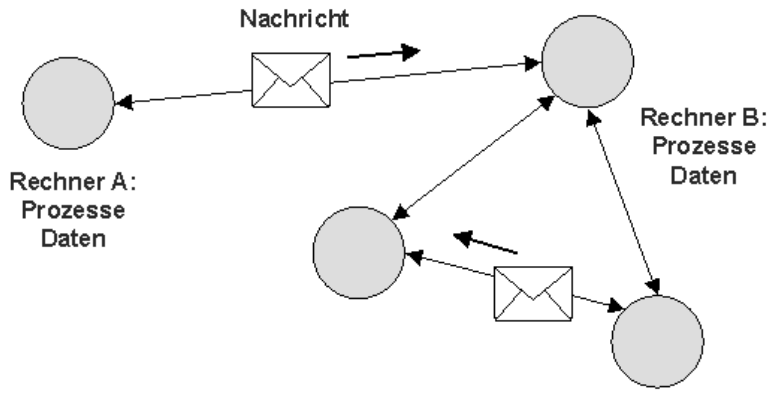

Abbildung 3.9: Aufbau eines verteilten Systems mit Austausch von Nachrichten.

Merkmal	**Zentrales System**	**Verteiltes System**
Wirtschaftlichkeit	gering	hoch
Verfügbarkeit	gering	hoch
Skalierbarkeit	schlecht	gut
Komplexität	gering	hoch
Konsistenz	einfach	schwierig
Sicherheit	hoch	gering
Technologie	einheitlich	heterogen

Tabelle 3.1: Vor- und Nachteile verteilter Systeme gegenüber zentralen Systemen.

Transparenz

Um die Komplexität eines verteilten Systems für den Anwendungsprogrammierer und den Anwender zu verbergen, werden im allgemeinen Forderungen nach Transparenz hinsichtlich bestimmter Merkmale gestellt. Das Ziel ist dabei, die Verteiltheit gewissermaßen durchsichtig zu machen und das System als zentralisiertes erscheinen zu lassen. Dadurch wird der Umgang mit der Verteiltheit eines solchen Systems erleichtert. Die wichtigsten Kriterien für die Transparenz gefordert wird, sind:

- *Ortstransparenz*: Der physische Ort einer Komponente muß nicht bekannt sein, um auf sie zugreifen zu können.
- *Zugriffstransparenz*: Auf lokale und entfernte Komponenten wird auf identische Weise zugegriffen.

- *Ausfalltransparenz*: Für den Benutzer ist der Ausfall einer Komponente transparent.
- *Technologie-Transparenz*: Die Verwendung von unterschiedlichen Technologien wird vor dem Anwender verborgen.
- *Concurrency-Transparenz*: Beim Zugriff mehrerer Benutzer auf eine Komponente ist diese gemeinsame Nutzung nicht erkennbar.

Die Realisierung dieser Transparenzkriterien macht einen großen Teil der Komplexität der in verteilten Systemen benötigten Mechanismen aus.

Client/Server-Modell

Durch das Client/Server-Modell werden zwei verschiedene Rollen eingeführt, die von Komponenten bzw. Prozessen in einem verteilten System eingenommen werden können. Dabei wird ein Mechanismus zur Kommunikation vorrausgesetzt, der synchron und auftrags-orientiert ist, d.h. der eine Nachricht sendende Prozeß bleibt inaktiv, solange er keine Antwort auf seine Auftrag bekommen hat. Diese Art des Kommunikationsmechanismus wird auch als *Remote-Procedure-Call* (RPC) bezeichnet und unterscheidet einen Sender, einen Empfänger, eine Auftragsnachricht und eine Ergebnisnachricht (siehe [CDK94]).

Die Rollen des Clienten im Client/Server-Modell ist die eines Dienstnutzers, der einen RPC auslöst und damit den Server auffordert, die bei ihm implementierte Prozedur mit den übergebenen Eingabeparametern auszuführen. Der Server als Diensterbringer sendet dann die Antwortnachricht mit den Ergebnissen der Prozedur an den Auftraggeber zurück. In verteilten Systemen, die nach dem Client/Server-Modell arbeiten, können Prozesse auch gleichzeitig die Rolle eines Clienten bzw. eines Servers annehmen, wenn z.B. ein Server eine Anfrage an einen untergeordneten Server weiterleitet.

Bei Verwendung dieses Kommunikationsmodells können verschiedene Arten von Fehlern auftreten. Zum einen können Nachrichten bei der Übertragung verlorengehen, zum anderen kann es zu Ausfällen von Prozessen durch Absturz kommen. Man unterscheidet zwischen Verlust von Auftragsnachricht bzw. Ergebnisnachricht und Ausfall von Server- bzw. Client-Prozeß. Je nachdem welche Dienstgüte von einer Anwendung erwartet wird, werden verschiedene Vermeidungsstrategien gegenüber diesen Fehlern benötigt. Idealerweise sollte eine Anwendung erwarten, daß auch beim Auftreten von Fehlern ein Auftrag genau einmal ausgeführt wird und genau ein Ergebnis zurückgeliefert wird (*Exactly-Once*-Strategie [VY99]). Dies einzuhalten ist in einem verteilten System jedoch nicht immer möglich. Für niedrigere Dienstgüteanforderungen existieren verschiedene andere Strategien, die zumindest einen Teilerfolg garantieren.

3.3.2 Verwendung eines Objektmodelles

Um die Komplexität eines verteilten Systems möglichst gut bewältigen zu können und effiziente Mittel zur Realisierung solcher Systeme zu erhalten, bietet sich die Verwendung eines Objektmodells für den Entwurf und einer objekt-orientierten Programmiersprache für die Entwicklung an. Im folgenden werden die Eigenschaften des Objektmodells und eine in objekt-orientierten Systemen verwendete Terminologie erläutert.

Zur systematischen Strukturierung eines allgemeinen Problemraumes beschreibt ein Modell, nach welchen Prinzipien die gewählte Darstellung bei der Modellierung erzeugt wird. Zur Strukturierung verteilter informationstechnischer Systeme, die schon durch ihre Größe und durch die hohen Anforderungen an ihre Funktionalität komplex sind, wird oft ein Objektmodell verwendet. Das

Objektmodell benutzt die Konzepte Abstraktion, Modularität, Kapselung und Hierarchie, um eine systematische Darstellung des verteilten Systems zu erlauben. Eine tiefgehende Diskussion dieser Begriffe und des Entwurfs objekt-orientierter Systeme ist in Booch et al. [BMN99] zu finden.

- *Abstraktion*: Eine Vereinfachung des Systems durch Hervorhebung relevanter Aspekte und Vernachlässigung nicht-relevanter Aspekte erlaubt die Reduzierung der Komplexität, indem eine grobere äußere Sicht verwendet wird. Die Auswahl der relevanten Aspekte für die Abstraktion ist dabei subjektiv. Außerdem liefert die Abstraktion eine Terminologie zur Beschreibung des Systems.

- *Modularität*: Mit Modularität ist die Unterteilung des Problembereichs bzw. Systems in Teilbereiche gemeint, die den relevanten Aspekten der Abstraktion entsprechen. Weiterhin werden damit wohl definierte Grenzen zwischen den Teilbereichen gefordert.

- *Kapselung*: Alle Details, die nicht als relevant für die äußere Sicht betrachtet werden, werden innerhalb der Bereichsgrenzen gekapselt. Durch die Kapselung wird eine innere Sicht auf die Implementierung eines Systembereichs ermöglicht und die äußere Sicht implementierungsunabhängig gehalten. Die Grenzen bzw. wohl definierten Schnittstellen werden zum Zugriff auf die gekapselten Details genutzt.

- *Hierarchie*: Dieses Konzept erlaubt die Beschreibung von Beziehungen zwischen verschiedenen Abstraktionen. Bekannte Hierarchien im Objektmodell sind die Spezialisierung (ausgedrückt durch *is-a*) und die Aggregation (ausgedrückt durch *part-of*) von zwei Abstraktionen. Spezialisierung bedeutet, daß eine Abstraktion alle Eigenschaften der zweiten Abstraktion besitzt (z.B. „Ein Hammer ist ein Werkzeug“). Aggregation bedeutet, daß eine Abstraktion sich bei ihrer Definition auf eine zweite Abstraktion beziehen kann (z.B. „Ein Stiel ist Teil eines Hammers.“).

Die Verwendung dieser vier Konzepte zur Darstellung eines Systems erlaubt seine Strukturierung, wodurch die Komplexität für den Betrachter reduziert wird. Die Modellierung ist abhängig von der subjektiven Auswahl der relevanten Aspekte, daher ist auf diese Weise keine kanonische Darstellungsweise möglich. Dies ist aber für den Entwurf und die Umsetzung eines verteilten Systems auch nicht notwendig gefordert.

Terminologie objekt-orientierter Systeme

In diesem Abschnitt werden die wichtigsten Begriffe erläutert, die im Umgang mit dem Objektmodell benötigt werden. Zuerst wird der Begriff des Objektes im Zusammenhang des Objektmodells diskutiert, danach wird auf den Austausch von Nachrichten und die Klassifizierung von Objekten nach Typen eingegangen.

- *Objekt*: Der Begriff des Objektes ist, wie der Name schon andeutet, zentral für das Objektmodell. Durch Anwendung der Konzepte Abstraktion, Modularität, Kapselung und Hierarchie wird das System als Menge von Objekten modelliert. Durch die Abgrenzung der Objekte mittels wohl definierter Schnittstellen wird die geforderte Modularität erreicht. Durch den Abstraktionsgrad wird die Granularität der Modellierung mit Objekten bestimmt.

  Jedes Objekt eines Systems hat eine eindeutige **Identität**, so daß es mittels einer Referenz adressierbar und zugreifbar ist. Weiterhin wird ein Objekt durch einen **Zustand** und ein **Ver-**

halten festgelegt, die zusammen die Funktionalität des Objektes definieren. Der Zustand entspricht in einer objekt-orientierten Anwendung den gekapselten Daten, während das Verhalten durch eine wohl definierte Methodenschnittstelle angegeben ist. In Abbildung 3.10 ist eine schematische Darstellung eines Objektes in einem objekt-orientierten Entwurf gezeigt, wie sie im weiteren Verlauf dieser Arbeit verwendet wird.

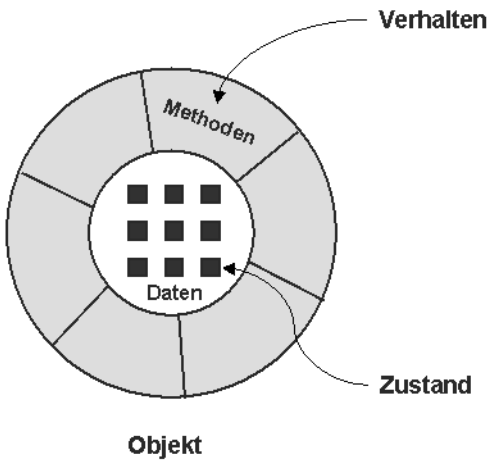

Abbildung 3.10: Schematische Darstellung eines Objektes mit Zustand und Verhalten.

Durch die nach außen festgelegte Funktionalität eines Objektes wird ein System unabhängig von einer bestimmten Implementierung. Solange eine wohl definierte Schnittstelle zur Verfügung gestellt wird, können andere Objekte über eine Referenz mit dem Objekt kommunizieren und Anfragen stellen, ohne die genaue Umsetzung im Inneren des Objektes kennen zu müssen.

- *Nachrichten*: Um eine Programmlogik umzusetzen, kooperieren verschiedene Objekte durch Austausch von Nachrichten. Diese Kommunikation erfolgt mit dem oben angesprochenen synchronen, auftrags-orientierten Mechanismus (RPC). Die Nachrichten richten sich an die Schnittstelle des gerufene Objektes und stellen Aufrufe von Schnittstellenmethoden dieses Objektes dar, mit denen der interner Zustand abgefragt oder geändert werden kann. Eine Nachricht besteht allgemein aus einem Selektor, der die Auswahl der angefragten Funktionalität angibt, und einer Reihe von Parametern, die als Eingabe für die aufgerufene Methode verwendet werden. Das ausführende Objekt bearbeitet die Anfrage und sendet eine entsprechende Nachricht mit den Ausgabeparametern zurück.

- *Klassen*: In einem Objektmodell wird eine Klassifizierung von Objekten vorgenommen, die Objekte mit gleicher Struktur und gleichem Verhalten zu einem Typ bzw. einer Klasse zusammenfaßt. Wenn ein Objekt die geforderten Eigenschaften eines bestimmten Typs erfüllt, so ist das Objekt eine *Instanz* dieses Typs. In objekt-orientierten Programmiersprachen werden Klassen definiert, die Struktur und Verhalten aller Objekte in dieser Klasse vorgeben. Durch Erzeugung einer Instanz der Klasse wird ein Objekt mit eindeutiger Identifizierung angelegt,

das einen bestimmten Zustand und das angegebene Verhalten aufweist. Die Instanzen in dem System tauschen dann Nachrichten aus, um eine Programmlogik zu realisieren.

3.3.3 Verteilungsplattformen für objekt-basierte Systeme

Zur Entwicklung und Ausführung verteilter Systeme in heterogenen Umgebungen sollten die oben genannten Konzepte und Forderungen umgesetzt werden, um die inhärente Komplexität handhabbar zu machen. Eine Verteilungsplattform bzw. *Middleware* unterstützt die Entwicklung und Anwendung, indem sie allgemein ausführbare Dienste für verteilte Systeme zur Verfügung stellt. Der Begriff Middleware bezieht sich darauf, daß es sich dabei um eine Software handelt, die zwischen Betriebssystem und der Anwendung arbeitet. In diesem Abschnitt sollen die Aufgaben und der Aufbau von Verteilungsplattformen für objekt-orientierte Systeme diskutiert werden. Objekte eignen sich durch ihre Eigenschaften besonders gut als Grundeinheiten der Verteilung. Der Kommunikationsmechanismus mittels Methodenaufrufe kann auch über Rechnergrenzen hinweg ausgeführt werden, indem die Objektreferenzen und Schnittstellen global bekannt gemacht werden. Nach einer allgemeineren Einführung wird die standardisierte Middleware CORBA [OMG98] als Beispiel herangezogen.

Folgende Aufgaben einer Verteilungsplattform sollen hier hervorgehoben werden (siehe auch [PR01]):

- *Unterstützung des Objektmodells*: Die im Objektmodell geforderten Konzepte sollen durch entsprechende Mechanismen der Verteilungsplattform unterstützt werden.
- *Operationale Interaktion*: Interaktion zwischen zwei Objekten soll entsprechend dem Methodenaufruf objekt-orientierter Programmiersprachen möglich sein.
- *Entfernte Interaktion*: Auch die Interaktion zwischen zwei Adreßräumen bzw. über Systemgrenzen hinweg soll unterstützt werden.
- *Verteilungstransparenz*: Es besteht aus Programmsicht kein Unterschied zwischen lokaler und entfernter Interaktion.
- *Technologieunabhängigkeit*: Unterschiedliche Technologien, wie verschiedene Programmiersprachen, Betriebssysteme, Rechnerarchitekturen und Netzwerke, sollen unterstützt und aus Programmsicht transparent gehalten werden.

Diese Forderungen werden umgesetzt, indem mittels der Verteilungsplattform eine horizontale Schicht zwischen Anwendung und Betriebssystem eingeführt wird. Unterhalb dieser Schicht wird durch die Middleware die Heterogenität der Systemtechnologie abgedeckt. Oberhalb der Schicht liegenden Anwendungen wird deren Funktionalität als *Application Programming Interface* (API) in Form einheitlicher Dienste zur Verfügung gestellt. Das API hängt von der Programmiersprache der jeweiligen Anwendung und der Middleware-Implementierung ab. Die Verteilungsplattform wird dem Entwickler als Programmbibliothek zur Verfügung gestellt.

Middleware am Beispiel von CORBA

CORBA stellt eine Standardisierung einer Verteilungsplattform zur Entwicklung und Anwendung von objekt-orientierten verteilten Systemen dar. Der Standard wird von der *Object Management*

Group (OMG) herausgegeben und weiterentwickelt. Der OMG gehören eine Vielzahl internationaler Mitglieder (über 800 im Jahr 2001) aus dem Bereich der Informationstechnologie an, die an der Standardisierung einer einheitliche Spezifikation für eine plattformunabhängige und interoperable Middleware mitwirken. Die beiden wichtigsten Standards der OMG sind die *Object Management Architecture* (OMA), eine allgemeine Plattform zur Beschreibung verteilter, objekt-orientierter Anwendungen, und die *Common Object Request Broker Architecture* (CORBA), die eine Spezialisierung der OMA und eine konkrete Verteilungsplattform beschreibt.

Die *Object Management Architecture* beinhaltet ein abstraktes Objektmodell und eine Referenzarchitektur. In Abbildung 3.11 ist der Aufbau der Referenzarchitektur mit seinen Komponenten gezeigt. Die Bezeichungen in den Abbildungen dieses Abschnitts sind in Englisch, der Sprache des Standards, angegeben.

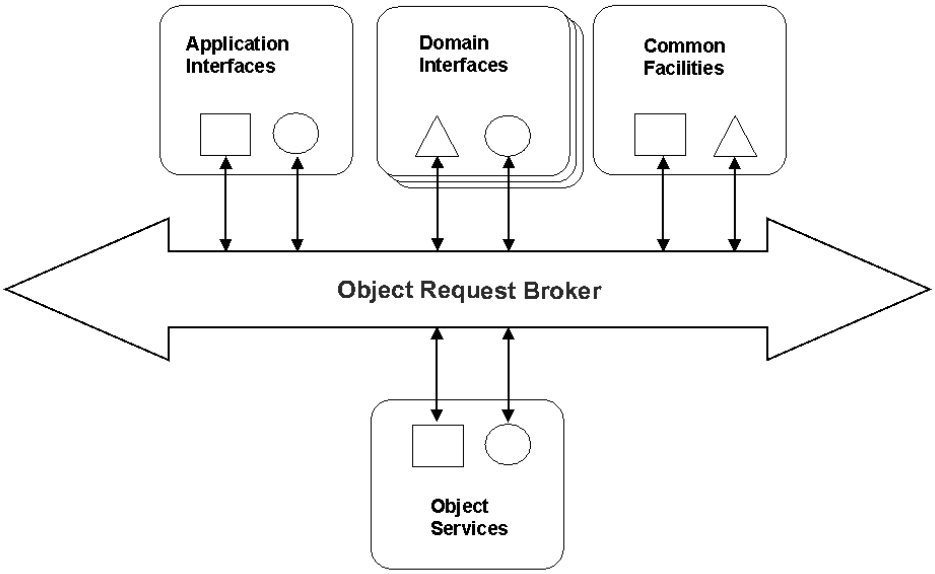

Abbildung 3.11: OMA-Referenz-Architektur.

Die OMA Referenzarchitektur beschreibt die Beziehungen zwischen den Objekten, die im OMA Objektmodell abstrakt definiert sind. Der Kern der Referenzarchitektur ist der *Object Request Broker* (ORB), über den die Objekte der verschiedene Kategorien kommunizieren. Folgenden Kategorien sind in der OMA beschrieben:

- *Objekt Services*: Hier werden horizontale Systemdienste festgelegt, die anwendungsunabhängig sind und häufig benötigt werden. Ein Beispiel ist der Namensdienst (*Naming Service*), der die Zuordnung von Namen und Objektreferenzen erlaubt. Weitere Dienste sind z.B. Vermittlungs- und Sicherheitsdienste.

- *Common Facilities*: Die Common Facilities stellen horizontale Anwenderdienste zur Verfügung, die in verschiedenen Anwendungskontexten häufig vorkommen (z.B. ein Druckdienst).

- *Domain Interfaces*: Hier werden vertikale Dienste zwischen verschiedenen Anwendungsbereichen festgelegt, wie etwa Medizin-, oder Finanzdienste.

- *Application Interfaces*: Die Application Interfaces sind anwendungsabhängige Dienste, die deshalb auch nicht von der OMG standardisiert werden.

Im CORBA-Standard wird das abstrakte Objektmodell der OMA konkretisiert, indem Basistypen (boolean, long, float,...), zusammengesetzt Typen (struct, union,...) und Signaturen zusammen mit einer Aufrufsemantik für die operationale Interaktion festgelegt werden.

Das Grundprinzip von CORBA wird durch kommunizierende Objekte in einer Client/Server-Architektur verdeutlicht. Ein CORBA Client-Objekt ruft einen Dienst eines CORBA Server-Objektes mit Hilfe des unterliegenden ORB auf. Dabei gilt das oben genannte Prinzip der Verteilungstransparenz für die Interaktionen. Der Server stellt eine Implementation der Objektfunktionalität des entfernten Objektes bereit, auf die mit Hilfe der Objektreferenz zugegriffen werden kann. Im CORBA-Standard wird dazu auf Client-Seite ein *Stub-Objekt* und auf Server-Seite ein *Skeleton*-Objekt mit Stellvertreterfunktion eingeführt. Der prinzipielle Mechanismus bei einem Aufruf eines entfernten Server-Objektes durch ein Client-Objekt mit CORBA ist in Abbildung 3.12 dargestellt.

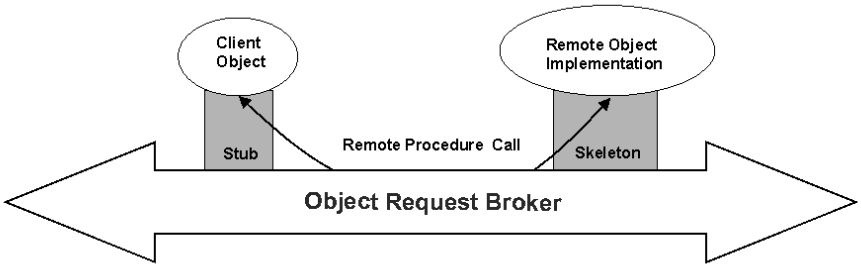

Abbildung 3.12: Aufruf von Methoden verteilter Objekte mit CORBA.

CORBA Komponenten

Die wichtigsten Komponenten des CORBA Standards werden nun anhand der Abbildung 3.13 vorgestellt. Alle zu CORBA gehörenden Komponenten sind dort grau dargestellt. Die Server- und Client-Objekte bzw. die Stellvertreterobjekte Stub und Skeleton gehören zur jeweiligen Anwendung. Die gestrichelte Linie deutet den Weg eines Aufrufs einer Methode der Server-Implementierung vom Client-Objekt an.

- *ORB-Kern*: Der ORB-Kern ist die zentrale Komponente von CORBA und überträgt Methodenaufrufe transparent zwischen Adreßraum- bzw. Systemgrenzen. Er findet Objektimplementierungen im Netzwerk und vermittelt, sendet und empfängt Daten. Außerdem stellt er dem Client und dem Server das ORB-Interface zur Verfügung. Im Zusammenhang der anderen Komponenten nimmt der ORB Operationen vom Aufrufadapter (Client-Seite) entgegen und liefert sie am Objektadapter (Server-Seite) wieder aus.

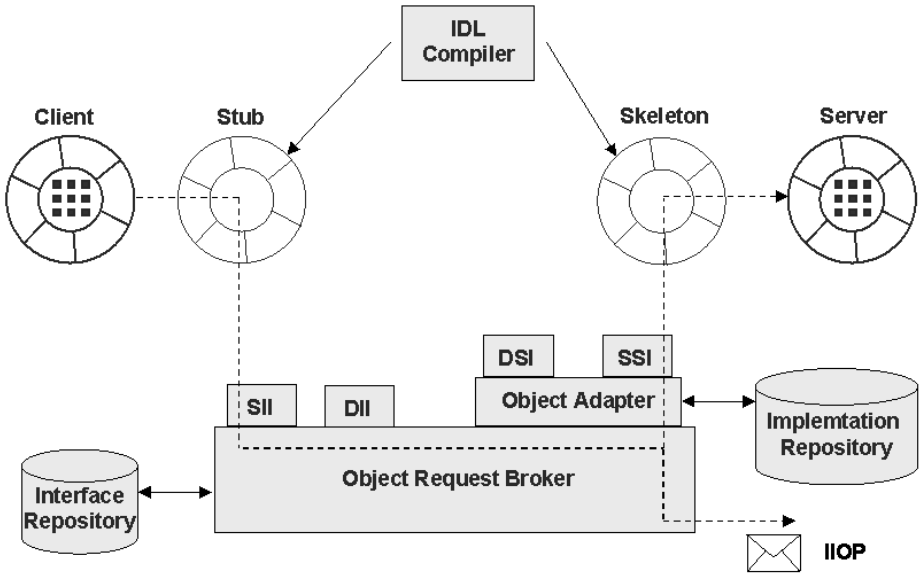

Abbildung 3.13: CORBA Komponenten.

- *IDL*: Die *Interface Definition Language* dient der Spezifikation von Objektschnittstellen. IDL ermöglicht die Trennung von Schnittstelle und Implementierung bzw. die Unabhängigkeit von Programmiersprachen, da sie eine eigene deklarative Sprache darstellt. Diese Sprache ist an C++ angelehnt und bietet neben der Typdefinition auch den Mechanismus der Schnittstellenvererbung an.

- *IDL-Sprachanbindung*: Durch Verwendung eines IDL-Compilers werden die IDL-Schnittstellen in die jeweilige Programmiersprache der Anwendung übersetzt. Die OMG definiert derzeit Sprachanbindungen für C, C++, Java, Smalltalk, Ada95 und COBOL. Aus einer Schnittstellendefinition werden vom IDL-Compiler stets zwei Stellvertreterobjekte Stub und Skeleton erzeugt. Der Stub dient dem Client als Proxy-Objekt für seine Aufruf und verhält sich wie das entfernte CORBA-Objekt. Das Skeleton-Objekt befindet sich auf dem Server und leitet den Aufruf an die Methodenimplementierung weiter. Die erzeugten Stub- und Skeleton-Objekte werden noch einmal von dem Compiler der verwendeten Programmiersprache übersetzt.

- *Aufrufadapter*: Die Client-Objekte übergeben ihre Aufrufe an den Aufrufadapter entweder über das Stub-Objekt oder direkt. Der ORB-Kern nimmt vom Aufrufadapter den jeweiligen Aufruf entgegen. Es werden zwei Aufrufadapter unterschieden. Das *Static Invocation Interface* (SII) dient zur Übergabe von Aufrufen mit Hilfe des vom IDL-Compiler erzeugten Stub-Objektes. Das *Dynamic Invocation Interface* (DII) wird dem Programmierer zur direkten Verwendung angeboten, um von vorübersetzten Stubs unabhängig zu sein.

- *Objektadapter*: Auf Server-Seite dient der Objektadapter als Verbindung zwischen ORB-Kern und Objektimplementierung. Er verwaltet den Lebenszyklus von Objekten und sorgt

für die Ausführung ankommender Aufrufe. Es werden wieder zwei Arten von Objektadaptern unterschieden. Das *Static Skeleton Interface* (SSI) dient der Anbindung an den Objektadapter mit Hilfe der vorgenerierten Skeleton-Objekte. Das *Dynamic Skeleton Interface* (DSI) erlaubt Methodenaufrufe von Objekten, deren Schnittstellen bei der Entwicklung noch nicht bekannt waren.

- *Interface Repository* und *Implementation Repository*: Das Interface Repository stellt eine Datenbank der Typinformationen bereit, mit dem CORBA-Objekte Schnittstelleninformationen zur Laufzeit erfragen können. Das Implementation Repository stellt analog eine Datenbank für Objektimplementierungen zur Verfügung.

Die Zusammenarbeit verschiedener CORBA-Produkte wird im Standard über die Inter-ORB Protokolle garantiert. Der CORBA-Standard gibt ein auf TCP basierendes Protokoll an, das dort *Internet Inter-ORB Protocol* (IIOP) genannt wird. Mit dieser vertikalen Schnittstelle ist die volle Interoprabilität von CORBA gewährleistet. Allerdings kann es trotzdem zu Kommunikationsproblemen zwischen verschiedenen ORB-Implementierungen kommen, da nicht alle Hersteller sich um volle Interoperabilität bemühen. Weitere Details zur Realisierung von Anwendungen mit CORBA werden in Kapitel 7 beschrieben. Für weitergehende konzeptuelle Informationen zum CORBA-Standard wird auf [Sieg96] verwiesen.

3.4 Verarbeitung von Bildinformationen

In diesem Abschnitt werden Methoden zur Bildcodierung und Übertragung in verteilten Visualisierungsanwendungen diskutiert. Diese Methoden werden für die hier vorgestellte Software-Architektur benötigt, da mit Objekt-Repräsentationen gearbeitet wird, die im Spektrum zwischen geometrie-basierten und bild-basierten Informationen angesiedelt sind.

Zur Übertragung von Bilddaten in verteilten Netzwerkumgebungen werden Techniken benötigt, die eine Reduzierung der Informationsmenge ohne merklichen Verlust an Bildqualität ermöglichen. Die Verfahren zur Kompression digitaler Bilddaten werden in zwei Gruppen eingeteilt, in verlustfreie (“lossless”) und verlustbehaftete (“lossy”) Methoden. Die Kompressionsraten liegen für die verlustfreien Methoden bei maximal 2:1 und können heute für verlustbehaftete Codierung bis zu 100:1 bei guter Qualität betragen, wobei das Verhältnis zwischen Originaldatenmenge und der Datenmenge nach der Kompression gemeint ist.

In Abbildung 3.14 sind Datenraten von gebräuchlichen Videoformaten und die Übertragungsraten heutiger Netztechnologie gegenübergestellt. Es ist zu erkennen, daß schon für das CIF-Format (352x288 Pixel) mit 30 Bildern pro Sekunde, 8 Bit Farbtiefe und mit 4:2:2 Abtastung (d.h. die Farbwerte U und V sind mit einer vierfach geringeren Rate abgetastet als die Luminanz) eine Datenrate von 36 MBit/s erreicht wird. Diese Datenrate kann in einem lokalen Netzwerk nur mit Fast-Ethernet oder ATM erzielt werden, während im Internet ein garantierte Bandbreite von dieser Größenordnung heute nicht zu erwarten ist. Aus diesem Grund wurden zur Übertragung von Bilddaten Codierverfahren entwickelt, die eine starke Reduzierung der Datenmenge ohne wahrnehmbaren Qualitätsverlust erlauben. Verfahren, die dabei hauptsächlich genutzt werden sind die Diskrete-Cosinus-Transformation (DCT), die Quantisierung von DCT-Koeffizienten, die bewegungskompensierte Prädiktion, die Entropiecodierung und als neueres Verfahren die Wavelet-Codierung. Viele Codierstandards, wie MPEG-1, MPEG-2, MPEG-4 oder H.263, arbeiten mit mehreren Techniken gleichzeitig und werden deshalb als Hybrid-Codierverfahren bezeichnet. Eine gute Beschreibung der heute genutzten Verfahren ist in [ES98] zu finden.

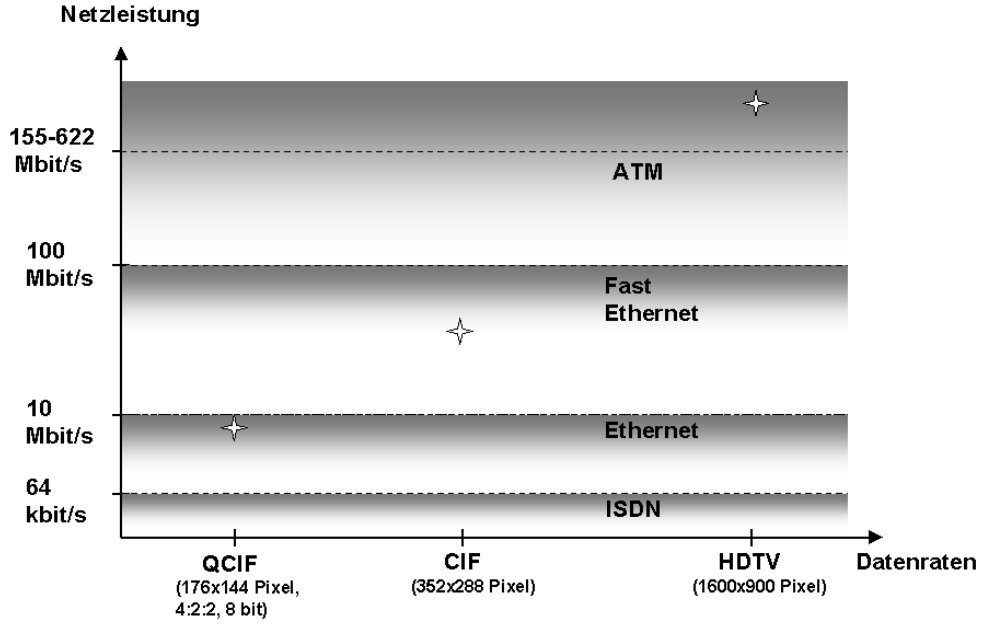

Abbildung 3.14: Gegenüberstellung von Datenraten und Netzleistung für digitale Videoformate.

3.4.1 Bilddaten Codierung der ersten und zweiten Generation

Das methodische Vorgehen von Codierverfahren kann auf abstrahierte Weise dargestellt werden, indem zwei aufeinander folgende Schritte benutzt werden. Wenn ein Bild in grundlegenden Einheiten, sogenannte Nachrichten (*Messages* [EK98]), eingeteilt wird, deren Kombination oder Überlagerung die ganze Bildinformation wiedergibt, dann kann das methodische Vorgehen als Extraktion und Codierung dieser Grundeinheiten beschrieben werden. In Abbildung 3.15 ist diese Beschreibungsweise anschaulich dargestellt.

Codierstandards der ersten und zweiten Generation können nun unterschieden werden, indem die Grundeinheit bei der Verarbeitung näher betrachtet wird. Bei der Codierung mit MPEG-1 ist diese Grundeinheit gegeben durch die Pixel selbst zusammen mit ihrer Bewegung in der Bildsequenz, die als numerische Werte aus dem Bild extrahiert und dann weiterverarbeitet werden. Die Algorithmen, wie DCT oder Bewegungsschätzung, arbeiten auf diesen Einheiten und das Bild ist eine Kombination dieser Einheiten in Form eines Rechtecks. Weitere Standards, die auf diese Weise arbeiten sind MPEG-2 (ISO), H.261 und H.263 (ITU).

Bei Videocodierstandards der zweiten Generation sind nicht mehr Pixel die Grundeinheiten bei der Extraktion und Codierung, sondern es werden nun zusammenhängende Objekte eine Bildes als grundlegende Messages aufgefaßt. Auf diese zweidimensionalen Bildobjekte werden dann die weiteren Codiermethoden angewandt. Ein Objekt wird beschrieben durch seinen Umriß, seine Textur und seine Bewegung. Durch Kombination bzw. Überlagerung wird auf der Decoderseite das ursprüngliche Bild wieder rekonstruiert. Diese Vorgehensweise hat neben der Möglichkeit zur höheren Kompression der Daten (z.B. nur ein Bewegungsvektor pro Objekt) auch den Vorteil, daß sie mehr der menschlichen Wahrnehmung und Informationsverarbeitung im Gehirn entspricht

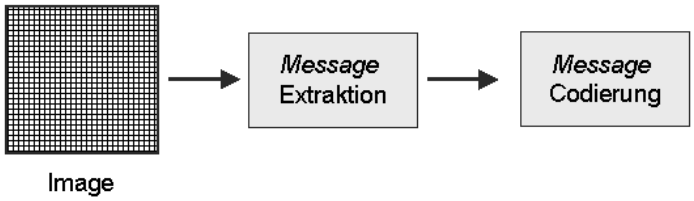

Abbildung 3.15: Message-Extraktion und -Codierung bei der visuellen Datenkompression.

und damit ein größeres Potential für zukünftige Anwendungen besitzt. Ein noch relativ junger internationaler Codierstandard, der zur zweiten Generation gehört, ist der ISO-Standard MPEG-4. MPEG-4 ([ISO99a]) wurde entwickelt, um multimediale und interaktive Inhalte auf Objektbasis synchronisiert zu übertragen. Der Objektbegriff wurde erweitert, indem neben Bildobjekten auch Audioobjekte und 3D Objekte in codierten Szenen enthalten sein können. Der Standard spricht daher von audio-visuellen Objekten. Die weiteren Komponenten des MPEG-4 Standards neben der Audio/Videocodierung dienen der optimalen Übertragung und Synchronisation dieser audiovisuellen Objekte. Im Rahmen dieser Arbeit wird jedoch vor allem der Videocodierstandard von MPEG-4 benutzt werden.

Schema eines Hybridcoders und -decoders der ersten Generation

Die Arbeitsweise eines Hybridcodecs (Coder und Decoder) soll anhand des MPEG-1 Videostandards erläutert werden. In Abbildung 3.16 ist der Coderaufbau schematisch dargestellt. Der Coder besteht aus den Komponenten für die DCT-Transformation, Quantisierung (plus jeweils deren inverse Transformation), Bewegungsschätzung, Bewegungskompensation und dem Entropiecoder. Hinzu kommen Bildzwischenspeicher und Komponenten zur Subtraktion und Addition von Bildinformationen. Am Eingang des Coders werden die digitalen Bilder sequentiell in Form von Pixelblöcken eingelesen. Am Ausgang werden die entropiecodierten Informationen über den codierten Bildinhalt, die Bewegungsvektoren und der jeweilige Codiermodus, also *Inter* oder *Intra*, als Bitstrom zur Übertragung ausgegeben. MPEG verwendet drei Frametypen, nämlich *I-Frame* (Intra-Modus), *P-Frame* (Inter-Modus) und *B-Frame* (Inter-Modus), die sich darin unterscheiden, ob sie Information aus vorhergehenden bzw. nachfolgenden Bilder zur Rekonstruktion benötigen. Eine Bildsequenz ist als Abfolge dieser Frametypen mit einer anfänglich festgelegten Ordnung angegeben.

Wird ein Bild im Intramodus codiert, so wird sein Inhalt zuerst DCT-transformiert und quantisiert, und das Resultat dann über den Entropiecoder ausgegeben. Gleichzeitig wird das Bild intern wieder rücktransformiert und dann in einem Bildspeicher abgelegt. Das darauffolgenden Bild sei nun inter-codiert, d.h. zum Beispiel ein P-Frame. Für dieses aktuelle Bild wird nun eine Bewegungsschätzung durchgeführt, wobei das zwischengespeicherte I-Frame als Referenzbild benutzt wird. Wenn diese Schätzung erfolgreich ist, werden die Bewegungsvektoren zusammen mit dem resultierenden Vorhersagefehler über den Entropiecoder ausgegeben. Auf Decoderseite wird dann dieses aktuellen P-Frame bestimmt, indem auf das zwischengespeicherte I-Frame die Bewegungs-

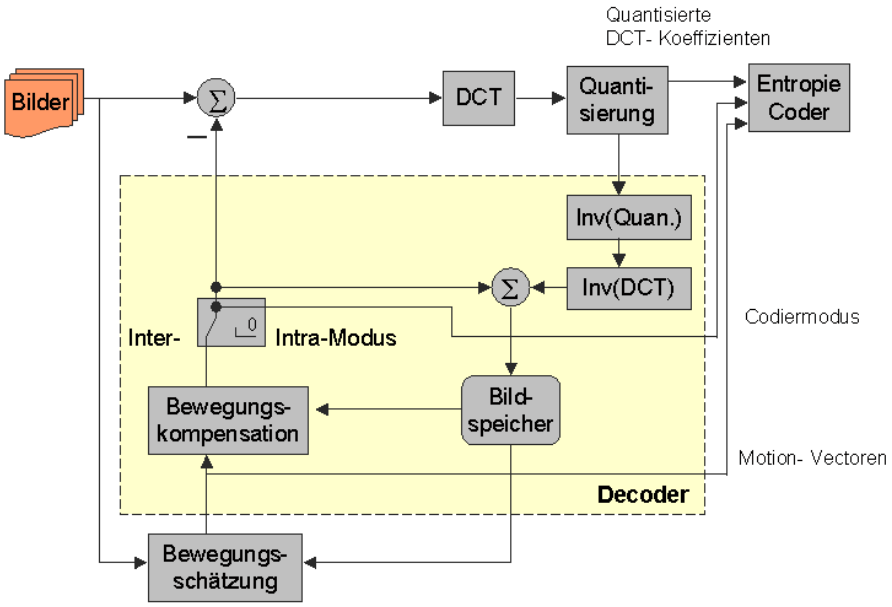

Abbildung 3.16: Schema eines MPEG-1 Videocoders.

vektoren und der Fehler angewandt werden. Im Falle von B-Frames wird diese Methode ähnlich angewendet, nur daß hier zwei I- bzw. P-Frames als Referenzbilder benutzt werden, wodurch die Berechnung komplexer, jedoch die Kompressionsrate erhöht wird. Für alle Frame-Typen gilt, daß durch entsprechende Anpassung der Quantisierung die Bitrate des Ausgabestroms geändert werden kann und die Bildqualität sich auf diese Weise optimieren läßt.

Durch nähere Betrachtung der Komponenten in Abbildung 3.16 wird klar, daß der Decoder vollständig im Aufbau des Coders enthalten ist (siehe markierten Bereich) [HH95]. Um wirklich nur den komprimierten Codierfehler übertragen zu müssen, wird, wie oben beschrieben, die benötigte Information durch sofortiges Decodierung des vorhergehenden Bildes, anschließende Bewegungskompensation und durch Vergleich mit dem aktuellen Bild im Coder berechnet. Da der arbeitsaufwendigste Teil, nämlich die Bewegungsschätzung, auf der Coderseite erfolgt, wird das zeitlich assymetrische Verhalten des MPEG-Codecs verständlich. Der Decoder erfüllt lediglich die weniger aufwendige Aufgabe, nach der inversen Entropiecodierung, Quantisierung und DCT-Transformation die benötigte Bildinformation im Zwischenspeicher zu halten. Anschließend werden die Bewegungsvektoren und der jeweiligen Fehler auf ein zwischengespeichertes Bild addiert, um das decodierte Bild zu erhalten.

Ein objekt-basierter Videocodierstandard der zweiten Generation: MPEG-4

Die Standards der ersten Generation sind block-basiert, d.h. die verwendeten Algorithmen arbeiten auf Blöcken von Pixeln des digitalisierten Bildes. Ein Bild ist dann eine Kombination von Pixelblöcken in Form eines Rechtecks. In Videocodierstandards der zweiten Generation, wie MPEG-4, werden Bilder nun als Kombinationen aus verschiedenen Bild- oder Szenenobjekten gesehen. Jedes Objekt bezieht sich auf eine Bildregion mit beliebiger zweidimensionaler Fläche und Kontur

innerhalb des rechteckigen Gesamtbildes. Der MPEG-4-Videocodec codiert für jedes dieser Objekte (VO) die Textur-, Bewegungs- und Konturinformation. Die Videoobjekte einer Szenen werden in elementare Bitströme codiert, mit einem Multiplexer zusammengelegt und als multiplexter MPEG-4-Gesamtstrom übertragen. Beim Decodieren werde dann Kombinationen dieser Objekte mit Hilfe der Konturinformation durch α*-Blending* schrittweise übereinandergelegt, um das Gesamtbild zu erhalten.

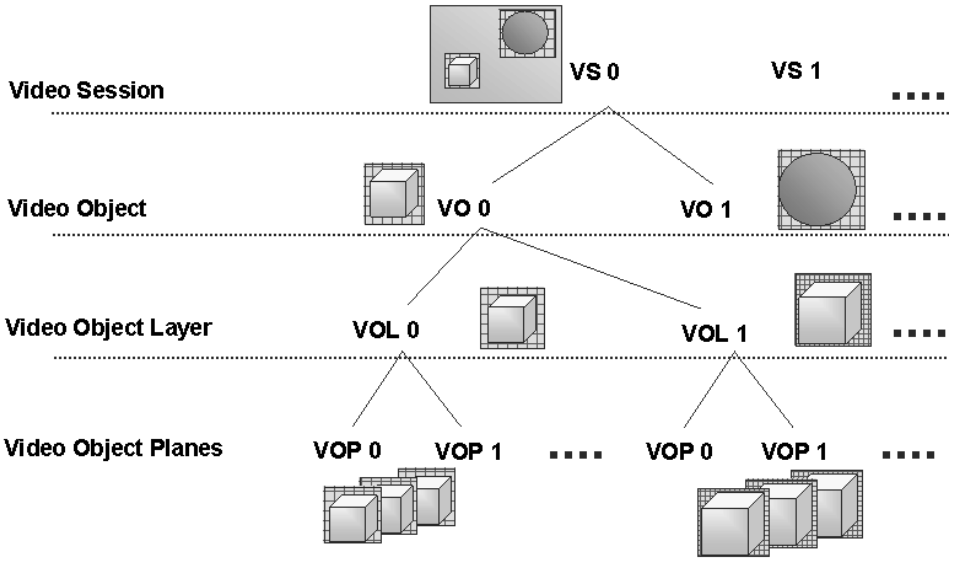

Abbildung 3.17: Struktur eines MPEG-4 Bitstroms.

Die Struktur des Gesamtbitstroms ist objekt-basiert und in verschiedenen hierarchische Schichten gegliedert. In Abbildung 3.17 ist die MPEG-4- Bitstromstruktur für eine Videoszene mit zwei Objekten dargestellt. Die Schicht mit der höchsten Ordnung ist die *Video Session (VS)*, die alle Objekte und deren Zuordnung in der Szenen beschreibt. Darunter folgt die Objektschicht *Video Object (VO)* mit den eigentlichen Objekten. Als nächstes folgt die Schicht *Video Object Layer (VOL)*, die ein Objekt in verschiedenen Auflösungen beschreibt. Die unterste Ebene bildet die *Video Object Plane (VOP)*-Schicht, die die eigentliche codierte Bildinformation für ein Objekt enthält. Bei der Übertragung einer MPEG-4 Szene wird zuerst die Szeneninformation übermittelt und dann werden die codierten Inhalte als gemultiplexte Elementarströme gesendet.

Neben Videoobjekten können MPEG-4 Szenen auch Audioobjekte und 3D Objekte enthalten. Die Verbindung zwischen diesen Objekten und der binären Szenenbeschreibung (BIFS) wird durch die Objektdeskriptoren (OD) gewährleistet. In Abbildung 3.18 sind der Aufbau eines MPEG-4 Stromes und die Vorgänge bei der Decodierung veranschaulicht. Nach dem Demultiplexen wird die Szenenbeschreibung zusammen mit den Objektdeskriptoren dazu verwendet, die decodierten Objekte in einer Szene anzuordnen. Darauf werden die 3D Objekte gerendert und diese Bildinformation zusammen mit den Videoobjekten als Gesamtszene angezeigt.

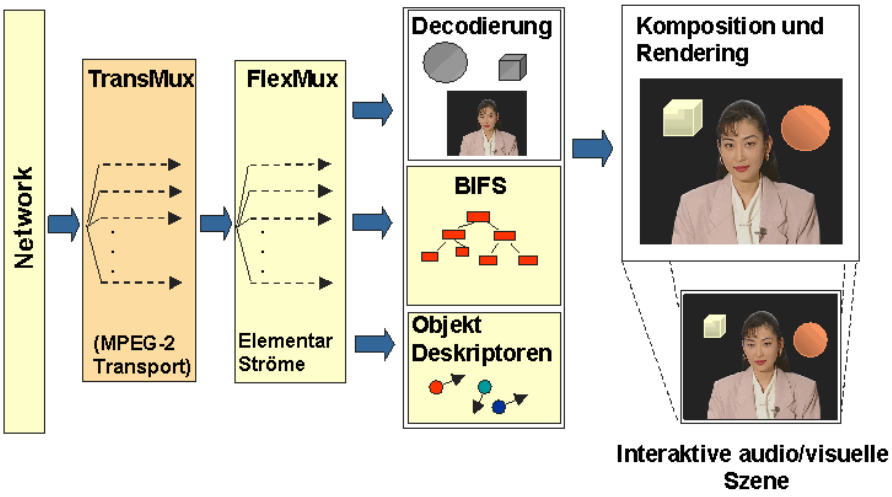

Abbildung 3.18: Decodierung eines MPEG-4-Bitstroms: Rekomposition und Rendering.

3.4.2 Netzwerkübertragung von codierten Bilddaten

Die Übertragung von digitalen Bilddaten in codierter Form geschieht in der hier untersuchten Software-Architektur in einer eigene Schicht. Die Bilddaten werden nicht als Aufruf- oder Rückgabeparameter eines entfernten Methodenaufrufs über eine Middleware übertragen, sondern als Bitstrom mit Hilfe der Protokolle TCP bzw. UDP über Sockets gesendet bzw. empfangen.

TCP ist ein verbindungs-orientiertes gesichertes Transportprotokoll, mit dem im Voll-Duplex-Betrieb und bei gesicherter, unveränderter Reihenfolge Ströme von Bytes ausgetauscht werden können. Die Adressierung erfolgt über IP-Adressen (IP: *Internet Protocol*) und eine Anzahl von logischen Ports. Die Verzögerungszeit, die dadurch entsteht, daß verlorene Pakete mit TCP noch einmal gesendet werden, macht dieses Protokoll für die Bilddatenübertragung mit Echtzeitanforderungen ungeeignet. Für Einzelbilddaten, die garantiert fehlerfrei beim Empfänger ankommen sollen, ist TCP jedoch dem UDP Protokoll vorzuziehen.

Zur schnellen Übertragung von Bildsequenzen wird ein verbindungsloser ungesicherter Transportmechanismus benötigt, bei dem allerdings auch Daten verlorengehen können. Mit dem *User Datagram Protocol* (UDP) wird ein solches Protokoll angeboten. UDP setzt auf dem IP-Protokoll auf und unterstützt das Multiplexing von Datagrammen zur Übertragung zwischen Rechnern im Internet. Neben der Erstellung und Kontrolle von Checksummen bietet UDP keine weiteren Dienste. Vor allem Multimedia-Anwendungen mit Echtzeitanforderungen nutzen das UDP-Protokoll, um beim Benutzer einen kontinuierlichen visuellen Eindruck erzielen zu können. Die Paketierung und Behandlung von Fehlern bei der Übertragung muß dabei an höhere Schichten der Anwendung weitergegeben werden. Als weiteres Protokoll, das höhere Funktionalitäten wie z.B. *Streaming* unterstützt, sei hier das *Real-Time Transport Protocol* (RTP) genannt. Eine weiterführende Diskussion zu Multimedia-Datenübertragung wird z.B. von Steinmetz [Stei00] gegeben.

In Abbildung 3.19 ist ein Szenario zur Übertragung von codierten Bilddaten mittels TCP bzw. UDP im Internet gezeigt. Die digitalen Bildaten werden mit einem Bildcoder komprimiert und dann an den Bild-Server übergeben. Der Bild-Server enthält einen internen Daten-Buffer, um Da-

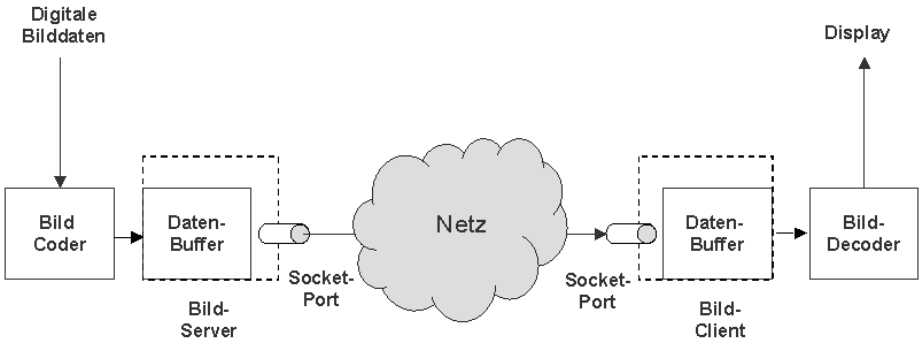

Abbildung 3.19: Netzwerkübertragung von codierten Bilddaten.

tenverluste zu vermeiden. Über eine Socket-Verbindung zum Benutzerrechner wird der codierte Bitstrom im Netz verschickt und auf Client-Seite von einem Bild-Client eingelesen. Beide Seiten verwenden die Socket-Implementierung ihrer jeweiligen Programmiersprache. Am Client werden die Bilddaten über einen Daten-Buffer an den Bild-Decoder weitergegeben, der als Ausgabe die zurücktransformierten Bilddaten an einen Display weiterleitet. Bei der Decodierung entstehen gewisse Informationsverluste, wenn kein Lossless-Verfahren angewendet wurde. Für Bildsequenzen finden in der Regel solche Lossy-Codierverfahren Anwendung, die versuchen, die Informationsverluste im nichtwahrnehmbaren Bereich zu halten und eine hohe Kompressionsrate zu ermöglichen.

Zur Versorgung mehrerer Benutzer mit gleichen Bilddaten werden Mechanismen verwendet, die entweder am Server eine Vervielfältigung der Datenströme in mehrere gleiche Ströme vornehmen oder es werden Multicast-Methoden benutzt. In diesem Fall werden die Bildaten an alle Teilnehmer einer Multicast-Gruppe versendet, ohne daß eine Datenvervielfachung nötig wird. Multicast-Gruppen müssen in diesem Fall allerdings von der unterliegenden Infrastruktur unterstützt werden (siehe [MB94]).

Kapitel 4

Konzepte zur verteilten 3D Visualisierung

In diesem Kapitel soll ein Überblick über den Stand der Forschung auf dem Gebiet der 3D Visualisierungskonzepte gegeben werden, die zum verteilten Arbeiten mit 3D Dokumenten eingesetzt werden. Einführend werden die Komponenten eines interaktiven 3D Visualisierungssystems in verteilten Umgebungen beschrieben. In diesem Zusammenhang werden auch wichtige Techniken zur Verteilung und zur Synchronisation von Szenengraphen diskutiert. Danach werden Konzepte zur Verteilung einer Visualisierungs-Pipeline vorgestellt, d.h. Visualisierungskonzepte für einen einzelnen Benutzer in einem verteilten System. Darunter fallen vor allem lokales und Remote-Rendering in einer Client/Server-Umgebung. Anschließend werden Mehrbenutzersysteme mit mehreren Visualisierungs-Pipelines diskutiert, die interaktive 3D Visualisierung zur Unterstützung der Kooperation von Benutzern einsetzen. Zum Abschluß des Kapitels werden Verfahren behandelt, die lokale und Remote-Visualisierung von 3D Dokumenten kombinieren.

4.1 Eine Klassifizierung verteilter 3D Visualisierungsysteme

Eine geeignete Klassifizierung für verteilte Visualisierungssysteme wird zum Beispiel von Brodlie et al. [BDG98] angegeben. Tabelle 4.1 gibt die verwendete Klassifizierung wieder: Unter kooperativer Visualisierung wird hier die synchronisierte 3D Visualisierung durch mehrere Benutzer verstanden, wobei die bearbeiteten 3D Dokumente ganz oder zum Teil übereinstimmen. Im folgenden werden Systeme besprochen, die immer auf mehrere Rechner verteilt sind und die von einem (verteilte Visualisierung) bzw. mehreren Benutzern (verteilte kooperative Visualisierung) verwendet werden. Die Verteiltheit kann sich daher auf die Daten, auf die Prozesse der Visualisierungs-Pipeline oder auf die Benutzer beziehen.

4.2 Komponenten verteilter 3D Visualisierungsysteme

Wie in Abschnitt 3.2 besprochen wurde, können interaktive 3D Visualisierungssysteme durch ein Modell bestehend aus einer Rendering-Pipeline und einer Aktions-Pipeline beschrieben werden. Hinzu kommen in realen Systemen noch Komponenten zum Einlesen bzw. Verteilen von Geometriedaten, zur Übertragung bzw. Darstellung von Bilddaten und zur Übertragung von anderen Daten, wie Benutzeraktionen oder Sicherheitsinformationen.

Anzahl Rechner ——— **Anzahl Benutzer**	**1**	**M**
1	Zentrale Visualisierung	Verteilte Visualisierung
N	Kooperative Visualisierung	Verteilte Kooperative Visualisierung

Tabelle 4.1: Klassifizierung von verteilten Visualisierungssystemen.

In diesem Anschnitt wird es darum gehen, die Komponenten eines Visualisierungssystems auf eine verteiltes System abzubilden. Dafür werden die wichtigsten Bestandteile in abstrakter Form als symbolische Komponenten dargestellt. Für die weitere Darstellung werden diese Symbole verwendet, um den Aufbau verschiedene konzeptioneller Systemansätze zu beschreiben. In Abbildung 4.1 ist ein Visualisierungssystem mit lokalem Rendering unter Verwendung einer Rendering-Pipeline auf dem Client-Rechner auf diese Weise gezeigt. Die wichtigsten Komponenten für eine allgemeine Beschreibung von 3D Visualisierungssystemen sind:

- die *3D Dokumente* selbst,
- die *3D Rendering-Pipeline*,
- eine *2D Abbildungskomponente*,
- eine *Komponente zur Interaktion* mit den 3D Daten,
- eine *Synchronisationskomponente* für verteilte Szenengraphen,
- eine *Systemgrenze* zwischen Rechnern.

Für all diese Komponenten sind Symbole in Abbildung 4.1 festgelegt, mit denen Visualisierungssysteme modelliert werden können. Die interne Funktionalität der Komponenten ist dabei gekapselt. Für die Komponenten Interaktion und Synchronisation von verteilten Szenengraphen werden die verwendeten Techniken in den Abschnitten 4.2.2 und 4.2.3 diskutiert. Außerdem wird in Abschnitt 4.2.1 die dynamische Verteilung von Szenengraphen besprochen. Einzelheiten zu der internen Umsetzung der anderen Komponenten wurden in Abschnitt 2.2 (*Digitale Dokumente*) und Kapitel 3 (*Methoden*) beschrieben.

Die verteilte Umgebung wird hier durch ein Client/Server-System repräsentiert, in dem mindestens ein Server-Rechner und ein Benutzerrechner existieren, die durch ein Netzwerk verbunden sind. Die Bereiche von Server- und Client-Rechner sind schraffiert dargestellt. Es wird vorausgesetzt, daß sich die 3D Dokumente, also die Geometriedaten, anfänglich auf dem Server befinden. Außerdem gibt es Möglichkeiten zum Retrieval und zum Zugriff auf die Dokumente, die es dem Benutzer erlauben, mit den Dokumenten zu arbeiten. Die Systemgrenze gibt in den Darstellungen jeweils den Bereich zwischen beiden Rechnern mit den Rechnergrenzen und dem verbindenden Netzwerk an. Im wesentlichen findet in diesem Bereich eine Übertragung von Dokumentdaten (Geometrie-, Bild-, Sicherheitsdaten) und von Synchronisationsdaten (Middleware-Kommunikation) mit Hilfe geeigneter Netzwerkprotokolle statt. In den weiteren Darstellungen mit

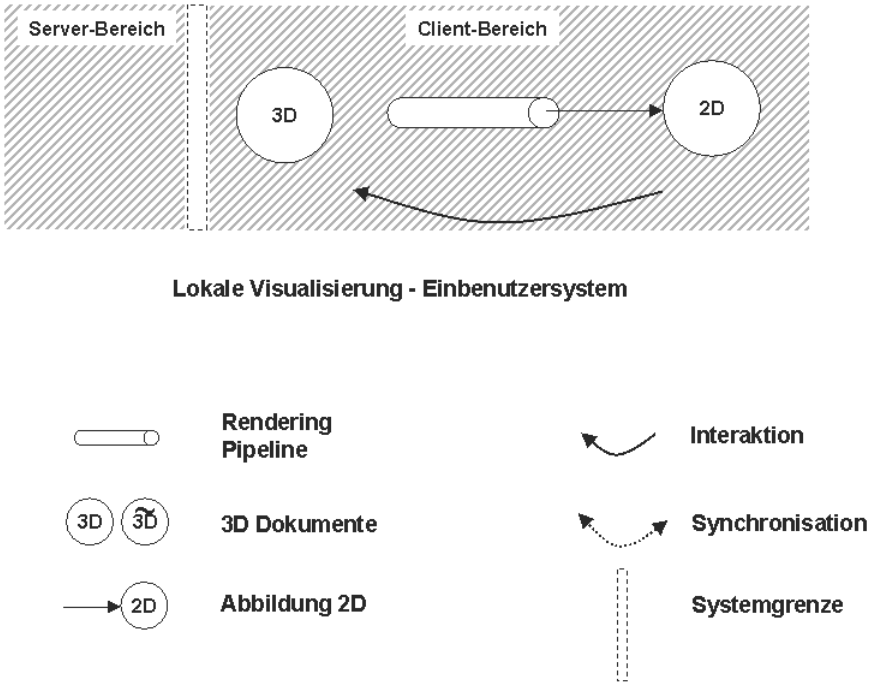

Abbildung 4.1: Symbolische Beschreibung von Komponenten eines verteilten Visualisierungssystems mit lokalem 3D Rendering für einen Benutzer.

den angegebenen Symbolen werden die Bereiche von Server- und Client-Rechner zur Vereinfachung nicht mehr explizit dargestellt, sondern durch eine Systemgrenze repäsentiert.

Die 3D Visualisierung wird als eine Abbildung der 3D Informationen auf visuelle 2D Informationen beschrieben. Dies wird mit den symbolischen Komponenten dargestellt, indem eine Rendering-Pipeline mit der 3D Szene als Input verbunden wird und der Bilddaten-Output mittels Rasterung, Übertragung und Darstellung am Ausgabegerät visualisiert wird (siehe Abbildung 4.1). Die 2D Abbildung wird auf diese Weise von den vorhergehenden Prozessen der Rendering-Pipeline getrennt, um später bei den Verfahren, die lokale und Remote-Visualisierung kombinieren, eine konsistente Darstellungsweise zu erlauben.

4.2.1 Verteilung von 3D Szenen

Da sich die 3D Dokumente im Falle eines möglichen Digital-Library-Szenarios anfänglich auf dem Server befinden, benötigt eine Software-Architektur zum verteilten Arbeiten Möglichkeiten zur Verteilung von 3D Dokumenten. In diesem Abschnitt sollen Methoden zur Umsetzung der notwendigen Funktionalität erläutert werden.

Auf Server-Seite werden die 3D Dokumente entweder in einer externen Datenbank vorliegen oder bereits im Szenengraph einer Remote-Visualisierungs-Anwendung integriert sein. In beiden Fällen soll der Benutzer durch eine einfache Auswahlaktion, die er zum Server schickt, in die Lage versetzt werden, ein 3D Dokument oder Teile davon in seine lokale Visualisierungsanwendung zu

integrieren. Dazu muß jedoch am Server zuerst auf die ausgewählten Daten zugegriffen werden und eine Übertragung in einem bekannten Datenformat über eine Systemgrenze hinweg stattfinden. Die einzelnen Schritte werden nun im Detail beschrieben. In Klammern hinter jedem Punkt ist angegeben, ob der Client oder der Server die Aktion ausführen.

- *Selektion* (Client): Je nachdem ob sich das gewünschte 3D Dokument in einer externen Server-Datenbank oder als Szenengraph in einer synchronisierten Remote-Visualisierungsanwendung befindet, wird der Benutzer entweder über eine WWW-Schnittstelle auf die externen DB (Fall A) oder über eine Selektionsmethode seiner lokalen Visualisierungsanwendung und eine Schnittstelle auf den Remote-Szenengraphen zugreifen (Fall B). Im Fall A werden entsprechende Retrieval-Daten (Metadaten oder inhalts-basierte Merkmale) an den Server geschickt und dort ausgewertet. Im Fall B wird in der Regel die Position auf dem Bildschirm beim Auslösen der Selektionaktion zum Server übertragen.

- *Retrieval* (Server): Die zum Server gesendeten Daten werden nun dort zum Retrieval verwendet. Im Fall A werden mit Hilfe eines *SQL-Select*-Befehles, mit dem die Retrieval-Daten an das DBMS weitergegeben werden, die passenden Dokumentdaten aus der Datenbank ausgelesen. Im Fall B wird mit Hilfe der Positionsdaten von der Remote-Visualisierungsanwendung eine *Pick*-Aktion (siehe hierzu [Wer98]) auf ihrem Szenengraph ausgeführt. Dabei wird davon ausgegangen, daß die Visualisierung am Server mit der am Client synchronisiert ist, d.h. das der Blickpunkt und -winkel zu jeder Zeit übereinstimmen. Das Resultat der Pick-Aktion ist ein Szenengraphpfad, der nun in einen internen Puffer oder in eine server-seitige Datei geschrieben wird. Die Ausgabe erfolgt dabei in einem Format, das zur Übertragung geeignet und auf beiden Seiten bekannt ist.

- *Übertragung* (Server oder Client): Hier kann zwischen einem *Push*-Verhalten, bei dem der Server selbstständig die am Client benötigten Daten dort bereitstellt, und einem *Pull*-Verhalten unterschieden werden, bei dem der Client die Informationen vom Server herunterlädt. Im Fall A handelt es sich um ein Pull-Verhalten, da das HTTP-Protokoll mit einem *Get*-Befehl von Client-Seite arbeitet. Die Übertragung wird auch von diesem Protokoll kontrolliert. Wenn die Übertragung zwischen zwei objekt-orientierten Visualisierungsanwendungen ablaufen soll, wie im Fall B, ist sowohl eine Pull-Verhalten als auch ein Push-Verhalten realisierbar. Die ausgewählten und in einem internen Buffer bereitliegenden Daten werden in Form von Byte-Feldern übertragen. Wenn ein Pull-Verhalten implementiert ist, wird vom Client mittels einer Middleware-Schnittstelle eine *Get*-Methode am Server aufgerufen, die als Rückgabeparameter das entsprechende Byte-Feld übergibt. Bei einem Push-Verhalten wird umgekehrt vom Server die *Set*-Methode eines entfernten Objekts am Client aufgerufen, das in diesem Fall ein Middleware-Server-Objekt darstellt. Die *Set*-Methode enthält dann als Eingabeparameter die ausgewählten Daten als Byte-Feld.

- *Lokale Integration* (Client): Am Client müssen die Daten nun von der Visualisierungsanwendung eingelesen werden. Durch den Aufruf einer entsprechenden Methode werden sie daraufhin in Form eines neuen Szenengraphen in die Anwendung integriert oder als Sub-Szenengraph in eine bestehende Szene eingefügt.

4.2.2 Interaktion mit verteilten Szenengraphen

Die 3D Interaktion mit Hilfe eines objekt-orientierten Visualisierungs-API geschieht über Benutzerschnittstellenereignisse, im weiteren *User Interface Events* genannt, und die Übertragung der

Benutzeraktionen auf die interne Szenendatenbank, also den Szenengraph. Jeder Knoten des Szenengraphen enthält Zustandsdaten (*Field Values* abgelegt in *Fields*) und stellt Methoden zum Zugriff auf die Daten bereit. Um mit einer 3D Szene zu interagieren, d.h. während der laufenden Visualisierung Manipulationen an der Szene vorzunehmen, kann der Benutzer diese Methodenschnittstelle nutzen und die entsprechenden Felder eines Objektes ändern. Als Beispiel sei ein einfaches Kugelobjekt genannt, das in einem Zustandsfeld seinen Radiuswert enthält. Durch Aufruf einer Methode $setRadius(value)$ des Kugelobjektes bzw. $setValue(value)$ bei direktem Zugriff auf das Feld kann der Benutzer den Radius interaktiv ändern und das Resultat in der Visualisierung sofort sehen.

Um nun auf entfernte Szenengraphen in einer verteilten Umgebung interaktiv (*Remote Interaction*, siehe hierzu auch [ESE99]) zugreifen zu können, müssen die benötigten Methoden des Objektes über eine Middleware-Schnittstelle zugänglich gemacht werden. Dazu werden die auf Client-Seite erzeugten Events zur Server-Anwendung übertragen und lösen dort die entsprechende Aktion aus. Dies wird mittels eines *Callback*-Mechanismus, der bei einer Selektion ausgelöst wird, und eines Middleware-Methodenaufrufes vom Client zum Visualisierungs-Server realisiert. Im Fall der interaktiven Manipulation von 3D Objekten werden durch eine Aktion die Feldwerte des entfernten Objektes geändert, die seine geometrischen Eigenschaften festlegen. Die Visualisierung wird dann automatisch von der Anwendung aktualisiert und es wird z.B. im Fall einer Remote Visualisierunsg-Architektur ein visuelles *Feedback* mittels der Bilddaten zum Client geliefert.

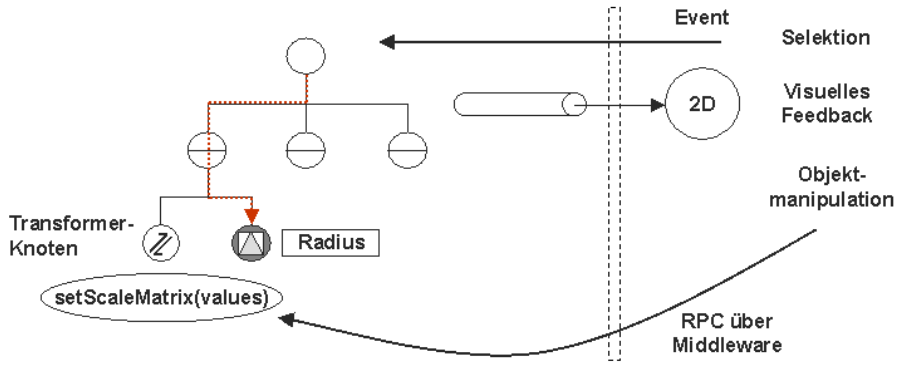

Abbildung 4.2: Interaktion mit einem entfernten Szenengraphen in einem verteilten System.

Mit dem folgenden Verfahren kann die Umsetzung der Remote-Interaktion vereinfacht werden. Der Zugriff auf die Felder entfernter Objekte setzt natürlich voraus, daß die Anzahl und Art der Zustandsfelder eines Objektes beim Zugriff erkannt werden. Ein Objekt müsste also bei jedem Zugriff identifiziert und einer Kategorie zugeordnet werden. Dies kann man umgehen, indem man zum Zeitpunkt der Selektion ein Transformer-Objekt in die Gruppe des selektierten Objektes an erster Stelle einfügt. Dieses Transformer-Objekt hat eine festgelegte Schnittstelle, mit der über einen entfernten Methodenaufruf sehr viele geometrische Transformationen mit einem Objekt durchgeführt werden können. Mit OpenInventor ist auf diese Weise z.B. die Skalierung, Rotation und Translation von beliebigen 3D Objekten möglich, die sich in der gleichen Gruppe hinter dem Transformer-Knoten befinden. Nachdem die Interaktion mit dem speziellen Objekt beendet

ist, wird das Transformer-Objekt wieder aus der Gruppe entfernt. In Abbildung 4.2 ist der Zugriff auf einen entfernten Szenengraphen mit Hilfe eines Transformer-Knotens gezeigt. Indem mit einem Middleware-Methodenaufruf (RPC, siehe Kapitel 3) auf die Felder des Transformer-Knotens zugegriffen wird, kann z.B. der Radius des in der gleichen Gruppe liegenden 3D Objektes durch Skalierung verändert werden. Um ein allgemein einsetzbares Verfahren für den Zugriff auf entfernte Dokumentobjekte zu ermöglichen, müssen die Schnittstellen dynamisch erzeugt werden und im System bekannt gemacht werden. Beliebige Parameter eines entfernten 3D Objektes werden so für die Interaktion zugänglich gemacht. In Abschnitt 9.3.2 von Kapitel 9 wird ein solches dynamisches Verfahren beschrieben, mit dem einem Benutzer auch die Synatx und Semantik einer entfernten Methode zur Parametermanipulation im Moment des Zugriffs auf ein Objekt verfügbar gamcht wird.

4.2.3 Synchronisation von verteilten Szenengraphen

Für kooperative Anwendungen, die mit 3D Visualisierung arbeiten, ist es notwendig, die Szenen mehrerer Clients zu synchronisieren. Dabei werden lokale Aktionen eines Benutzers über Systemgrenzen hinweg zu den anderen Benutzeranwendungen übertragen und dort in der gleichen Weise auf den Szenengraph angewendet. Zur vollständigen Synchronsiation von verteilten Szenengraphen werden sowohl die Kamerabewegungen eines Benutzers, als auch seine Objektinteraktionen mit Hilfe von Middleware-Methodenaufrufen auf den entfernten Szenengraph übertragen. Teilweise Synchronisation, bei der die Navigation jedes Benutzers unabhängig ist und lediglich alle Objektinteraktionen im ganzen System synchronisiert seien sollen, wird durch die Übertragung von Selektions-Events und die Aktualisierung der entsprechenden Feldwerte in den ausgewählten Objekten realisiert. Die Synchronisation verläuft dabei natürlich in beide Richtungen, so daß jeder Benutzer die Aktionen aller anderen Benutzer sehen kann.

In Abbildung 4.3 ist die vollständige Synchronisation zweier verteilter Visualisierungsanwendungen gezeigt, bei der ein aktiver Client seine Aktionen auf einen im Moment passiven Client überträgt. Die Kamerasynchronisation geschieht über ein Sensorobjekt im aktiven Szenengraph, das bei jeder Änderung des Kameraobjektes einen Event auslöst. Über einen *Callback* wird für dieses Ereignis eine Methode aufgerufen, die nun ihrerseits über eine Middleware-Schnittstelle die entsprechenden Parameter der entfernten Kamera aktualisiert. Im wesentlichen müssen zur Kamerasynchronisation die Position, der Blickwinkel und die Blickrichtung bekannt gemacht werden.

Zur Synchronisation von Objektinteraktionen werden über einen *Callback* die Event-Parameter, die bei der Selektion eines Objektes am aktiven Client ermittelt wurden, zur entfernten Anwendung übermittelt. Dort wird mit diesen Parametern die gleiche Aktion wie am lokalen Szenengraph ausgelöst. In diesem Fall ist dies einen Pick-Aktion, die die Pfadinformation des betreffenden Objektes im Szenengraph zurückgibt. Mit Hilfe dieser Pfadinformation werden dann die entsprechenden Felder des ausgewählten Objektes aktualisiert. In [HSF99] wird ein Namensschema für verteilte Szenengraphen vorgeschlagen, das die Identifikation von beliebigen Objekten in verteilten, aber übereinstimmenden Szenengraphen ermöglicht. Ein solches Schema zur Identifikation ist nötig, wenn eine teilweise Synchronisation vorliegt, d.h. wenn die Benutzer unabhängig voneinander navigieren können und daher die Bildschirmpositionen bei der Selektion nicht mehr übereinstimmen. Falls eine vollständige Synchronisation realisiert ist, kann die Identifikation einfacher über die Bildschirmposition des selektierten Objektes erreicht werden.

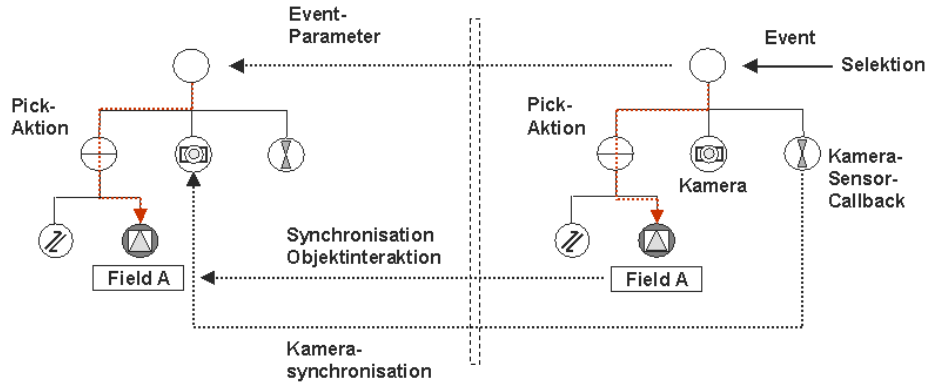

Abbildung 4.3: Vollständige Synchronisation zweier verteilter Szenengraphen über Middleware. Aktiver Client (rechts) und synchronisierter Client (links).

4.3 Konzepte zur Verteilung einer Visualisierungs-Pipeline

Wie in Kapitel 2 im Abschnitt 2.5 beschrieben, sind die wichtigsten Kriterien, die zur Unterstützung verteilten Arbeitens in 3D Visualisierungsumgebungen erfüllt werden sollten, der Schutz von Dokumenten, die Berücksichtigung der verfügbaren Übertragungsbandbreite zu einem Client-Rechner und die Berücksichtigung seiner lokalen Leistung. Bei der interaktiven Visualisierung wird zudem eine minimale Verzögerungszeit für das visuelle Feedback benötigt, um einen kontinuierlichen visuellen Eindruck beim Betrachter zu erreichen.

Im folgenden werden verschiedene Konzepte zur verteilten 3D Visualisierung vorgestellt und mit Beispielen aus der Literatur illustriert. Anhand der angegebenen Kriterien werden die Vorteile und Nachteile der einzelnen Konzepte für den Einsatz in verteilten Visualisierungsanwendungen diskutiert. Im Mittelpunkt stehen dabei offene, kooperative Informationsräume, d.h. Umgebungen vernetzter Rechner mit einem Client/Server-Ansatz, die von außen über eine Authentifizierung zugänglich sind und die Kooperation zwischen Benutzer erlauben. Offen bzw. von außen zugänglich meint hier, daß die Nutzung der angebotenen Dienste nicht auf einen vorher eingeschränkten und daher abgeschlossenen Bereich, wie z.B. ein Intranet, beschränkt ist. Dadurch werden Sicherheitsaspekte und die Berücksichtigung von Heterogenität des Informationsraumes besonders wichtig.

Der Prozeß der Visualisierung verläuft vom Einlesen der 3D Dokumente, über das 3D Rendering bis zur Darstellung der visuellen Daten z.B. auf einem Bildschirm oder mit Hilfe eines immersiven Displays. Die gebräuchlichsten Konzepte, die sich in der Verteilung der Visualisierungskomponenten unterscheiden, sind die lokale 3D Visualisierung und die Remote-Visualisierung in verteilten Systemen, die im folgenden beschrieben werden. Dabei befinden sich die 3D Dokumente anfangs auf dem Server und die Benutzer greifen von ihrem Rechner über das Netzwerk auf die entfernten Daten zu.

4.3.1 Lokale 3D Visualisierung mit Datenreplikation

Das wichtigste Merkmal dieses Konzeptes ist die Übertragung einer Kopie der 3D Dokumente auf den Client-Rechner, d.h. eine Replizierung der Daten. Die Visualisierungs-Pipeline wird danach vollständig auf dem lokalen Rechner ausgeführt und also nicht verteilt. Prinzipiell muß nach der Datenverteilung keine Verbindung zum Server gehalten werden, da auch die Interaktion mit den 3D Daten komplett lokal ausgeführt wird. Wie später erläutert werden wird, verlangen kooperative Anwendungen, die nach diesem Ansatz arbeiten, jedoch eine Synchronisation durch Übertragung von Aktionen zu den anderen Client-Anwendungen.

In Abbildung 4.1 vom Anfang dieses Kapitels ist eine Client/Server-Umgebung zur lokalen 3D Visualisierung mit Datenreplikation gezeigt. Nachdem die 3D Daten in den Client-Bereich übertragen wurden, findet dort das Einlesen in die 3D Rendering-Pipeline, das 3D Rendering selbst und die Ausgabe auf den Bildschirm statt. Die Interaktion wird durch die Anwendung von Aktionen auf die 3D Daten realisiert, indem z.B. die Feldwerte in den Zustandsfeldern eines beliebigen Objektknotens eines Szenengraphen verändert werden (wie in Abschnitt 4.2.2 beschrieben). Die Komponenten der interaktiven Visualisierung sind in Abbildung 4.1 durch die hier gewählten Symbole dargestellt.

Die Vorteile dieses Vorgehens sind zum einen die maximale Verfügbarkeit der 3D Daten für den Benutzer und zum anderen die minimale Verzögerung bei der Visualisierung. Beides ist gewährleistet, da die Daten lokal als Kopie vorliegen. Eine weitere Verminderung der Verzögerungszeit kann in der Regel auch nicht durch Ausnutzung einer leistungsstarken Server-Maschine zum Rendering erzielt werden, wie es bei der Remote-Visualisierung geschieht, da die Übertragung der Bilddaten durch ein Netzwerk immer eine relativ große Verzögerung bedeutet.

Der Ansatz lokaler 3D Visualisierung mit Datenreplikation bringt aber auch Nachteile mit sich. Zum einen wird für sehr komplexe Modelle die Übertragung der Geometriedaten selbst zum Problem, da eine entsprechend hohe garantierte Bandbreite z.B. im Internet nicht zur Verfügung steht. Wenn ein Benutzer in einer großen, zusammenhängenden 3D Welt navigiert, kann die Geometrieübertragung ein flüssiges Arbeiten unmöglich machen. Zum anderen kann die lokale Leistung und Kapazität des Client-Rechners für sehr komplexe Modelle nicht ausreichend sein. Dieser Punkt ist allerdings durch die heutige rasante Entwicklung der Prozessorenleistung und Speicherkapazitäten nicht in einem ähnlichen Maße kritisch, wie der vorhergehende Punkt. Schließlich wird die Replizierung von geschützten 3D Dokumenten in offenen Umgebungen zu einem entscheidenden Problem. Wenn zur 3D Visualisierung immer Kopien der Originaldaten verteilt werden müssen, dann können z.B. Urheberrechte nicht mehr ohne weiteres geschützt und Vertraulichkeitsanforderungen nicht erfüllt werden.

Ansätze und Software-Architekturen in der Literatur

In der Literatur sind einige Systeme beschrieben, die nach diesem Konzept der Datenreplikation arbeiten. In [HSF99] beschreiben Hesina et al. eine Software-Architektur bei der 3D Szenen als Kopien eines zentralen Szenengraphen auf dem Server verteilt werden und Updates über den Server an andere Client-Rechner in einer kooperativen Anwendung gesendet werden. Dort wird das Visualisierungs-Toolkit OpenInventor zur Realisierung verwendet und entsprechend zur Synchronisation von verteilten Szenengraphen erweitert. Schmalstieg und Gervautz beschreiben in [SG96] ein System, bei dem Geometriedaten adaptiv übertragen werden, d.h. daß nur die jeweils zum Rendering benötigte 3D Information am Client vorgehalten wird. In [HS98] beschreiben Hesina und Schmalstieg eine Software-Architektur, die diese angepaßte Geometrieübertragung in Internet-basierten virtuellen Umgebungen umsetzt. Dieser lokale Ansatz wird dort als *Remote Rendering*

bezeichnet, was jedoch nicht mit der weiter unter diskutierten Remote-Visualisierung verwechselt werden darf.

In Einzelbenutzeranwendungen wird der lokale Visualisierungsansatz sehr oft verwendet, indem die Daten einfach über einen WWW-Server zum Herunterladen bereitgestellt werden und die Visualisierung ganz dem Client überlassen wird. Diese Technik wird im offenen Informationsraum des WWW benutzt und durch die Angabe von *MIME-Types* (siehe [RFC1521]) unterstützt, die mit den Daten übertragen werden und angeben, welche Anwendung zur Darstellung aufgerufen werden soll. Eines der ersten Systeme dieser Art wurde von Ang et al. [AMD94] vorgestellt (siehe auch [RMW98]).

4.3.2 Remote 3D Visualisierung mit zentraler Datenhaltung

Bei diesem Konzept wird die Visualisierungs-Pipeline so verteilt, daß das Einlesen der Daten, das 3D Rendering und die Ausgabe von digitalen Bilddaten am Server geschieht. Die generierten Bilder des 3D Dokumentes werden dann in komprimierter Form zum Client-Rechner übertragen und dort dargestellt. In Abbildung 4.4 ist dies gezeigt, indem eine Trennung der 3D Rendering-Pipeline am Server und der 2D Abbildung im Client-Bereich über eine Systemgrenze hinweg vorgenommen wurde. Die Interaktion mit den 3D Daten geschieht vom Client-Rechner aus, indem die Aktionen wieder über die Systemgrenze hinweg zum Server übertragen und dort angewendet werden. Auf Client-Seite wird im Prinzip nur eine Anwendungssoftware benötigt, die Bilddaten interaktiv darstellen kann und die Benutzeraktionen geeignet aufnehmen und für die server-seitige 3D Rendering-Anwendung codieren kann.

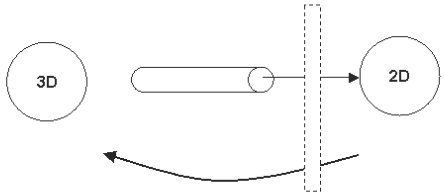

Abbildung 4.4: Remote 3D Visualisierung mit einer Rendering-Pipeline.

Die originalen 3D Daten bleiben bei diesem Konzept auf dem Server und sind dem Benutzer nicht direkt zugänglich. Für kooperative Systeme bedeutet dies eine zentrale Datenhaltung. Die Vorteile dieses Konzeptes ergeben sich direkt aus diesem Umstand, daß nun keine Originaldaten mehr als Kopie zum Client übertragen werden und damit der Schutz von Urheberrechten und die Erfüllung von Sicherheitsanforderungen erleichtert wird. Außerdem wird ein System auf Client-Seite unabhängig von speziellen Hardware- und Software-Anforderungen für das 3D Rendering, da dort lediglich 2D Daten angezeigt werden müssen. Dadurch ergibt sich eine höhere Flexibilität für den Einsatz auf einem weiten Spektrum von Endgeräten, das vom leistungsstarken PC, über Notebooks bis hin zu PDAs (*Personal Digital Assistant*) reichen kann. Über einen Rückkanal zum Server findet dabei die Interaktion mit den 3D Inhalten statt.

Der Hauptnachteil dieses Ansatzes ist die Verzögerungszeit, die durch das *Off-Screen-*

Rendering und die Codierung der Bilder am Server bzw. die Übertragung der komprimierten Bilddaten entsteht. Auch durch die Decodierung und Anzeige am Client entsteht natürlich noch einmal eine Verzögerung. Die Verzögerung durch Rückübertragung der Aktionen ist dagegen meist vernachlässigbar, da keine großen Datenmengen verarbeitet werden müssen. Auch die Anforderungen an die Auflösung der Bilddaten können zu einer unannehmbaren Verzögerung des visuellen Feedbacks führen. Bei diesem Konzept muß daher immer ein Kompromiß zwischen Bildqualität und Verzögerung angestrebt und berücksichtigt werden.

Ansätze und Software-Architekturen in der Literatur

Es wurden in den letzten Jahren mehrere Systeme zur entfernten 3D Visualisierung mit Hilfe von *Image-Streaming*-Techniken vorgestellt. Engel et al. [ESE99] stellen eine Remote Visualisierungs-Architektur für Volumendaten im medizinischen Bereich vor. Mittels einer verteilten CORBA-Anwendung und einer gepufferten Übertragung von Bilddaten stellen sie einem Benutzer die fast vollständige Interaktionsfunktionalität des OpenInventor API zur Verfügung, ohne daß die 3D Daten zum Client-Rechner übertragen werden müssen. Es werden verschiedene Bandbreiten in lokalen Netzen mit verschiedenen Kompressionverfahren getestet und die Bildqualität mit der Bildwiederholrate in Verbindung gebracht. Die erzielten Bildwiederholraten lagen bei 1 bis 7 Bilder pro Sekunde. Die dort vorgestellte Architektur erlaubt auch die Ausweitung des Systems auf mehrere Nutzer, die gemeinsam ein zentral abgelegtes 3D Modell visualisieren. Die Autoren sprechen auch den Nutzen des Ansatzes zum Schutz der Vertraulichkeit von medizinischen Daten an.

Haulsen und Jung [HJT98] stellen ebenfalls ein System zur Kontrolle von entfernten virtuellen Umgebungen über Bilddatenströme und einen ISDN-Rückkanal vor. Hintergrund sind hier kooperative Anwendungen zur 3D Visualisierung von Stadtentwicklungsdaten. Bei einer Bildqualität von $352 \star 288$ Pixel (CIF-Format) erzielen die Autoren eine Bildwiederholrate von maximal 4 Bildern pro Sekunde. Zur Bildcodierung wurde ein Wavelet-basierter Videocodec verwendet. Die Umsetzung und Untersuchung einer eigenen Remote-Visualisierungs-Architektur wird vom Autor der vorliegenden Arbeit in [Loe00a] beschrieben. Dort wird ein MPEG-4 Videocodec [Sik97] verwendet, um visuelle Objekteigenschaften in den Bilddatenströmen bei der Übertragung zu erhalten. Mit diesem System soll erreicht werden, daß Benutzer mit Hilfe unterscheidbarer 2D Objekte auf entfernte 3D Objekte interaktiv zugreifen können. Die Tests in der prototypischen Umsetzung ergaben, ähnlich wie in den oben genannten Systemen, eine hohe Verzögerung bei der Visualisierung. Die Bildwiederholraten lagen bei maximal 2 Bildern pro Sekunde bei einer Übertragungsdatenrate von 128 kBit/s.

4.4 Verteilte 3D Visualisierung für kooperative Anwendungen

Im vorhergehenden Abschnitt wurden Systeme für Einzelbenutzeranwendungen beschrieben, bei denen durch die Verteilung der Visualisierungsprozesse eine Kooperation auf Systemebene zwischen verschiedenen Rechnern stattfand. Hiefür wurde die Bezeichung *verteilte Visualisierung* benutzt. Brodlie et al. [BDG98] unterscheiden darüberhinaus eine Kooperation auf der Benutzerebene, d.h. eine Zusammenarbeit von mehreren Menschen, die ein verteiltes Visualisierungssystem verwenden. Dies wird hier in Anlehnung an [BDG98] als *Verteilte, kooperative Visualisierung* bezeichnet (vgl. Tabelle 4.1). In diesem Abschnitt soll es nun um solche Mehrbenutzersysteme zur verteilten 3D Visualisierung gehen. Im Prinzip kommen dabei die beiden oben genannten Konzepte der lokalen 3D Visualisierung mit Datenreplikation und der Remote 3D Visualisierung mit

zentraler Datenhaltung zum Einsatz. Als zusätzliche Komponente wird ein Mechanismus zur Synchronisation der verteilten 3D Daten und der verschiedenen Visualisierungs-Pipelines der einzelnen Benutzer eingeführt. Systemkonzepte zur Synchronisation während der 3D Visualisierung werden im folgenden diskutiert.

Im wesentlichen besteht ein verteiltes, kooperatives Visualisierungssystem aus mehreren verteilten Visualisierungs-Pipelines, die durch Austausch von 3D Daten und Steuerungsparametern synchronisiert werden. Das Ziel ist es, die Kooperation von Benutzern mit einer gemeinsamen Datenbasis durch ein synchronisiertes visuelles Feedback zu unterstützen. In Abschnitt 4.2.3 wurde zwischen einer vollständigen Synchronisation, bei der sowohl die Navigation in einer 3D Szene als auch die Manipulation an den Objekten zu jedem Zeitpunkt für jeden Benutzer übereinstimmen, und teilweiser Synchronisation unterschieden, bei der nur Objektmanipulationen angeglichen werden. Wenn auch Erzeugung und Löschen von Objekten in einer virtuellen Umgebung erlaubt sind, müssen in beiden Fällen Geometriedaten und Steuerungsparameter über ein Netz ausgetauscht werden.

In Abbildung 4.5 sind zwei Konzepte zur Realisierung von Mehrbenutzersystemen mit lokaler 3D Visualisierung und Remote 3D Visualisierung dargestellt. Im linken Teil der Abbildung ist ein System mit zwei Benutzern gezeigt, die durch Datenreplikation ein 3D Dokument auf ihren Rechner übertragen haben und nun jeweils eine lokale Visualisierungs-Pipeline benutzen (**Lokale Visualisierung - Multi-User**). Die Bilderzeugung, Bilddarstellung und Interaktion passiert also lokal. Um nun die Kooperation möglich zu machen, werden die Benutzeraktionen von Client 1 über die Systemgrenzen hinweg zu Client 2 übertragen und umgekehrt. Dort werden sie dann jeweils auf die 3D Daten angewendet. Im Ergebnis sieht jeder Benutzer alle Aktionen, die in dem verteilten System an den 3D Daten vorgenommen werden. In Abbildung 4.5 ist die Synchronisation als gestrichelter Pfeil mit zwei Spitzen, der über die Systemgrenzen hinweggeht, dargestellt.

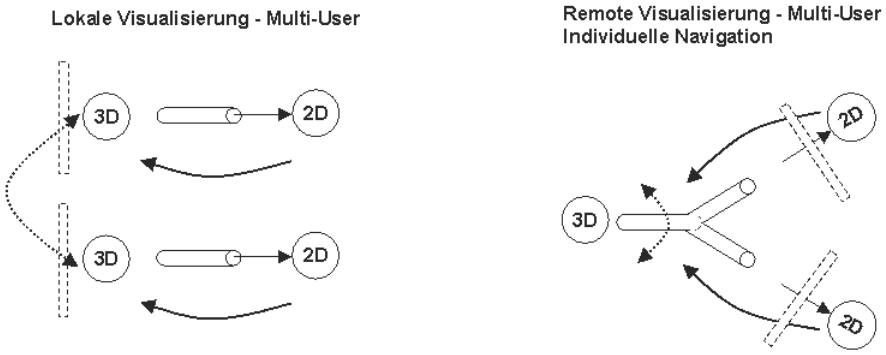

Abbildung 4.5: Multi-User-Visualisierung: Lokale Visualisierung (links) und Remote-Visualisierung mit individueller Navigation (rechts).

Um mit einem solchen System arbeiten zu können, muß nun eine Abstimmung der Aktionen auf einer höheren Ebene stattfinden. Dies kann z.B. durch Vergabe eines Tokens geschehen, der jeweils nur einem Benutzer erlaubt, die Daten zu verändern. Alle anderen Benutzer können dann eventuell zwar noch autonom in der Szene navigieren, aber keine Manipulationen der Objekte mehr vornehmen. Erst wenn der privilegierte Benutzer den Token freigibt, können andere Teilnehmer diesen

erhalten. Eine weitreichende Darstellung von Verfahren zur Aufrechterhaltung und Steuerung der Konsistenz von verteilten 3D Szenen in virtuellen kooperativen Umgebungen wird von Sighal und Zyda [SZ99] gegeben.

Um **Remote 3D Visualisierung** mit zentraler Datenhaltung in einer verteilten, kooperativen Anwendung nutzen zu können, müssen die Benutzeraktionen zum Server übertragen werden und dort synchronisiert auf das 3D Modell angewendet werden. Mit Hilfe der 3D Rendering-Pipeline auf dem Server werden Ansichten des 3D Modells erzeugt, die diese Aktionen visualisieren. Da bei der Remote Visualisierung lediglich Bilddaten zum Client übertragen werden, würden in diesem Fall alle Benutzer die gleiche Ansicht des 3D Modells als visuelles Feedback bekommen. Es lässt sich dabei also aus konzeptionell Gründen nur eine vollständige Synchronisation realisieren. Will man auch eine unabhängige Navigation zulassen, so muß die Rendering-Pipeline am Server vor der Kameratransformation aufgeteilt werden. Dadurch wird dann für jeden Benutzer aus derselben Datenbasis eine eigene Ansicht mittels seiner jeweiligen Kameraparameter erzeugt und zum Client-Rechner übertragen. Diese Methode der teilweisen Synchronisation bei der Remote Visualisierung kann mit einem Szenengraphkonzept durch periodisches, synchronisiertes Umsetzen der Kameraparameter erreicht werden. In Abbildung 4.5 auf der rechten Seite ist dieses Konzept mit autonomer Navigation der Clienten durch Aufteilung der Rendering-Pipeline in zwei Teil-Pipelines (**Remote Visualisierung - Multi-User - Individuelle Navigation**) gezeigt. Der Server-Rechner hat damit eine erhöhte Last, da er im Fall von zwei Benutzern in einer Zeiteinheit auch zwei Ansichten und die doppelte Bilddatenmenge erzeugen muß. Für N Benutzer erhöht sich die Last also maximal um das N-fache, wenn keine Redundanzen bei der Bilderzeugung genutzt werden. Wenn eine Untergruppe von n Benutzer eine gemeinsame Navigation wünscht und die anderen Benutzer autonom navigieren wollen, so lässt sich ein solches Szenario allerdings relativ einfach mit diesem Konzept umsetzen. Die Server-Last reduziert sich dementsprechend. Die in dieser Arbeit beschriebene Software-Architektur SCA3D nutzt das beschriebene Konzept, um einen server-seitigen Navigationsraum für die Visualisierung von 3D Dokumenten zu realisieren.

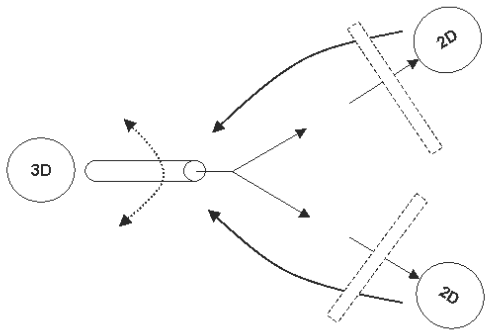

Abbildung 4.6: Multi-User Visualisierung: Remote-Visualisierung mit vollständiger Synchronisation.

In Abbildung 4.6 ist eine verteiltes, kooperatives Visualisierungssystem mit vollständiger Synchronisation gezeigt, das, wie oben beschrieben die einfachste Grundform des Remote Visualisierungskonzeptes darstellt. Benutzeraktionen werden am Server synchronisiert und auf das zentrale 3D Modell übertragen. Es handelt sich dabei um Kameraparameter und Manipulationsparameter

der einzelnen Benutzer. Aus diesen Daten und dem Szenengraph wird eine Bildansicht erzeugt und zu allen Benutzern übertragen.

Es wird schnell klar, daß ein solches Szenario nur mit einem privilegierten Benutzer, der die Navigation leitet, sinnvoll funktionieren kann. Für Gruppenanwendungen, wie z.B. virtuelle Klassenzimmer mit einem Leiter und einer Gruppe von Lernenden, kann dieses Konzept daher nützlich sein. Die Möglichkeit zur Multicast-Übertragung der Bilddaten und die zentrale Datenhaltung, die einen Schutz der Daten in offenen Informationsräumen erlaubt, sind für ähnliche Anwendungen von Vorteil.

Ansätze und Software-Architekturen in der Literatur

Verteilte Visualisierungsanwendungen für mehrere Benutzer wurden in den letzten Jahren intensiv untersucht und es existieren relativ viele experimentelle Systeme. Viele der Anwendungsfelder kommen aus den Bereichen Medizin, Computer-Aided-Design (CAD) und den Naturwissenschaften wie Geowissenschaften oder Chemie. Sehr viele der in der wissenschaftlichen Literatur beschriebenen Systeme arbeiten mit dem Konzept der lokalen Visualisierung mit Datenreplikation. Die meisten verwenden einen Client/Server-Ansatz, um die verteilten Datenbanken konsistent zu halten. Ein Beispiel hierfür ist COVISE [RFL98], das an der Universität Stuttgart entwickelt wurde. Ein weiteres System, AVANGO [GLG00] vom Institut für Medienkommunikation der GMD, arbeitet mit lokalen Replikationen und einem Szenengraphenkonzept (SGI Performer). Es ist für die Entwicklung echtzeit-fähiger Anwendungen im Bereich immersiver virtueller Umgebungen angelegt. Hesina et al. beschreiben in [HSF99] ihr System mit dem Namen *Distributed OpenInventor*, das auf dem OpenInventor API aufbaut und eine server-seitige Hauptkopie der verteilten Szene zur Synchronisation verwendet. Ein System, das für den Einsatz im Internet entwickelt wurde und zur Unterstützung der Kooperation von Benutzern auch Avatare benutzt, ist das DIVE-System von Carlsson et al. [CH93]. Es basiert nicht auf einem Client/Server-Ansatz, sondern verwendet ein *Peer-to-Peer*-Konzept, bei dem jeder Peer gleichberechtigt ist und Aktionen über Multicast-Gruppen versendet werden. Es existiert in diesem System also kein Server für zentrale Aufgaben.

Im Gegensatz zu der Vielzahl von Systemen mit lokalem Visualisierungskonzept sind Anwendungen eher selten, die reine Remote 3D Visualisierung mit zentraler Datenhaltung am Server verwenden. Die Hauptnachteile dieses Konzeptes, die sich aus der zentralen Datenhaltung und dem visuellen Feedback durch Bilddatenübertragung ergeben, sind eine relativ hohe Verzögerungszeit und die linear mit jedem zusätzlichen Benutzer ansteigende Server-Last, wenn autonome Navigation erlaubt sein soll. Trotzdem lohnt sich die Untersuchung dieses Ansatzes für den zukünftigen Einsatz in Anwendungen, die zum einen Sicherheitsanforderungen bezüglich geschützter 3D Daten an ein verteiltes Visualisierungssystem stellen und zum anderen die Möglichkeiten zur Skalierung der Belastung der Benutzerrechner benötigen. Daher sollen zwei Systeme vorgestellt werden, die das Konzept der Remote 3D Visualisierung für die Kooperation mit 3D Dokumenten nutzen.

Engel et al. [ESE99] beschreiben ein System zur Remote 3D Visualisierung mit Hilfe von server-seitig erzeugten Bilddatenströmen, die durch das Rendering von medizinischen Volumendaten entstehen. Die Aktionen von den Teilnehmern einer Visualisierungskonferenz werden über eine CORBA-Schnittstelle an den Server gesendet und dort auf einen OpenInventor-Szenengraphen angewendet. Die Übertragung der Bilddaten in codierter Form wird über UDP Socket-Verbindungen realisiert (siehe auch Abschnitt 3.4.2). Über den Server ist eine Datenbank mit den medizinischen Volumendaten zugreifbar, so daß mehrere Benutzer einen Datensatz auswählen und gemeinsam visualisieren können. Die Benutzer-Software wird dabei lediglich zur Anzeige der Bilddaten und zur Eingabe von Aktionen verwendet.

Ein prototypisches System zur Remote 3D Visualisierung über Bilddatenströme wird vom Autor in [Loe00a] vorgestellt. In Abschnitt 4.3.2 wurde das System schon kurz angesprochen. Das Konzept wird als *Visual Interaction Approach (VIA)* bezeichnet, da ein Benutzer über 2D Information mit angereicherter Objektinformation, wie Objektkonturen und -texturen, mit den server-seitigen 3D Objekten arbeitet. Auch hier können mehrere Benutzer zusammenarbeiten und dabei 3D Dokumente aus einer Datenbank visualisieren. Allerdings wurde keine Möglichkeit zur autonomen Navigation in das System integriert, da der Hauptaspekt in der Untersuchung der objekt-basierten visuellen Interaktion lag. Dadurch muß es einen privilegierten Benutzer geben, der die Navigation leitet. Das prototypische System wird in Kapitel 7 noch einmal kurz diskutiert.

4.5 Kombination von lokalem und Remote-Rendering für verteilte 3D Szenen

Um die Vorteile der lokalen und der Remote 3D Visualisierung in verteilten Anwendungen zu kombinieren und die Nachteile eines solchen Systems zu minimieren, werden in aktuellen Untersuchungen, die in diesem Abschnitt vorgestellt werden, beide Konzepte kombiniert. Um z.B. die Performanz eines spezialisierten High-End-Graphikrechners für sehr komplexe 3D Daten nutzen zu können, wird dieser über ein Netzwerk als Remote-Visualisierungs-Server verwendet. Damit der Benutzer aber gleichzeitig über eine hohe Bildwiederholrate bei der Manipulation von Einzelobjekten verfügt, werden diese Objekte lokal auf seinem Rechner gerendert und dann mit der server-seitige visuellen Information in einer Szene angezeigt. Dadurch wird es möglich, die Belastung eines Clients zu skalieren, 3D Dokumente flexibel zu schützen und die Netzlast an die verfügbare Bandbreite anzupassen. Zudem wird eine weites Spektrum von Synchronisationsmöglichkeiten für die verteilte 3D Visualisierung in kooperativen Anwendungen möglich.

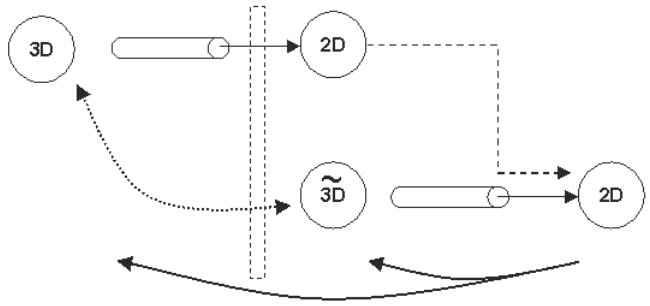

Abbildung 4.7: Kombination von lokaler und Remote-Visualisierung für einen Benutzer.

In Abbildung 4.7 ist eine Möglichkeit für ein kombiniertes Konzept anhand der in diesem Kapitel verwendeten Symbole gezeigt. Die 3D Szene, die sich anfangs auf dem Server befindet, wird während der Visualisierung in eine server-seitige Szene $\mathbf{3D}$ und eine client-seitige Szene $\mathbf{\tilde{3D}}$ aufgeteilt. Die Client-Szene enthält neben den übertragenen 3D Objekten für die lokale Interaktion auch noch eine 2D Projektionsfläche, auf die die server-seitige visuelle Information als Textur ab-

gebildet wird. Der Benutzer sieht auf diese Weise eine Szene, die aus einem Hintergrundbild vom Server und den lokalen 3D Objekten im Vordergrund zusammengesetzt ist. Durch Synchronisation der Visualisierungs-Pipelines auf dem Server und auf dem Client entsteht ein einheitlicher visueller Eindruck der Gesamtszene, die dem originalen 3D Dokument am Server entspricht. Die Benutzeraktionen werden sowohl auf die Client-Szene als auch auf die server-seitige Szene angewendet. Über die Systemgrenze hinweg werden dabei 3D Daten, Bilddaten und Kontrollparameter übertragen.

In Abbildung 4.8 ist das oben abstrakt dargestellte Konzept nun ausführlicher dargestellt. Die Ausgangsszene wird hierbei in eine Server-Szene und eine Client-Szene unterteilt. Durch Synchronisation der beiden Visualisierungs-Pipelines am Server und am Client entsteht ein visueller Eindruck der Gesamtszene. Der LoI-Regler (Level-of-Information) kontrolliert dabei die Verteilung der Szenen und die Repräsentationsform eines Objektes, das übertragen wird. Dieser Ansatz ist die Grundlage für das Konzept der Skalierbaren Szenen.

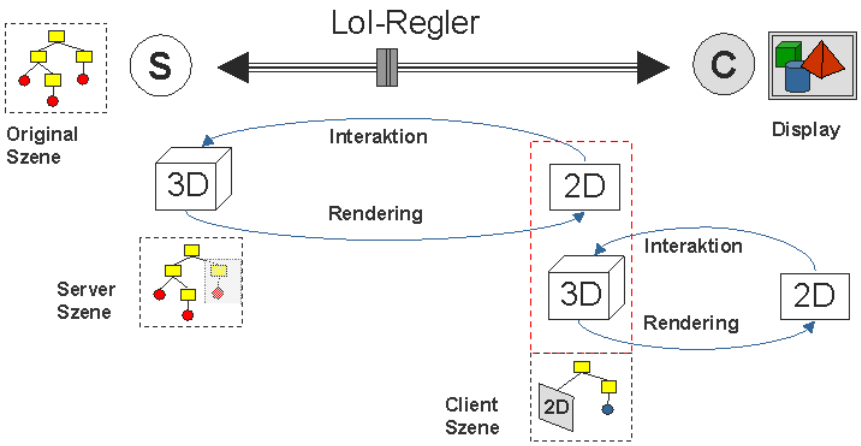

Abbildung 4.8: Kombiniertes SCA3D-Konzept mit dynamischer Verteilung der Originalszene.

Ansätze und Software-Architekturen in der Literatur

Kombinierte Ansätze sind aktueller Gegenstand der Forschung und sind in der Literatur noch nicht zahlreich vertreten. Hier sollen zwei Systeme erwähnt werden, die lokale und Remote 3D Visualisierung für den Einsatz in verschiedenen Bereichen kombinieren. Engel et al. [EHTE00] kombinieren server-seitiges mit client-seitigem Rendering, um komplexe medizinische Volumendaten für den breiten Einsatz im klinischen Umfeld interaktiv visualisieren zu können. Auf Server-Seite kommt ein OpenInventor-System zum Einsatz, während auf Client-Seite ein Java3D Renderer und ein Java2D Viewer verwendet wird. Mehrere Szenarien zum Einsatz in der Praxis werden diskutiert und die Entwicklung von kombinierten Anwendungskonzepten wird als sehr vielversprechend bezeichnet. Verteilte kooperative Visualisierung wird in dem Artikel nicht thematisiert.

Ein Konzept zur Navigation in entfernten virtuellen Umgebungen durch Bilddatenübertragung wird von Mann und Cohen-Or [MCO97] beschrieben. Dabei werden zur Reduzierung der benötigten Datenübertragung und der Bildaktualisierungsrate nur diejenigen Bildbereiche neu vom Server gesendet, die sich bei der Navigation stärker verändert haben. Am Client befindet sich lediglich

eine Geometriebasis, die zur vollständigen Visualisierung mit Texturbildern vom Server gerendert wird. Durch Übertragung von Referenzansichten und Extrapolation von Bildbereichen wird der visuelle Eindruck bei der Navigation am Client aufrechterhalten, bis aktualisierte Texturbilder vom Server geliefert werden. Dieser kombinierte Ansatz ist vor allem für komplexe virtuelle Welten auf einem Server von Vorteil, die entfernt visualisiert werden sollen, wobei die Netzlast gering gehalten werden muß. Die Erweiterung des Ansatzes für kooperative Anwendungen wird von den Autoren ebenfalls nicht nicht diskutiert.

Die Software-Architektur SCA3D, die in den folgenden Kapiteln vorgestellt und untersucht wird, stellt die dynamische Erweiterung eines kombinierten Ansatzes für verteilte kooperative Visualisierungssysteme dar und geht damit über den Stand der aktuellen Technik hinaus.

Kapitel 5

Verteilte Visualisierung mit Skalierbaren Szenen

Nachdem in den vorigen Kapiteln die Grundlagen, Methoden und der Stand der Forschung zum Thema verteilte Visualisierung von 3D Dokumenten erläutert wurde, wird nun ein eigenes Konzept vorgestellt, das mit einer Kombination von lokalem und Remote-Rendering arbeitet. Das Kapitel beginnt mit einer Einführung in das Prinzip der Skalierbaren Szenen. Dann wird eine Einordnung in den Stand der Forschung gegeben, gefolgt von einer detaillierten Beschreibung des Konzeptes mit seinen wichtigsten Komponenten. Im weiteren wird dann auf das Management von Sicherheitsinformationen und das Kooperationsmodell im Konzept der Skalierbaren Szenen eingegangen. Die Realisierbarkeit eines dynamischen Verhaltens der auf dem Konzept basierenden Software-Architektur wird im Anschluß diskutiert und eine Zusammenfassung schließt dann das Kapitel ab.

5.1 Einführung: Das Prinzip der Skalierbaren Szenen

Das *Konzept der Skalierbaren Szenen* stellt eine Lösung für die verteilte Visualisierung von geschützten 3D Dokumenten unter Berücksichtigung von Anbindung und Leistung der Benutzerrechner in heterogenen Umgebungen dar (siehe [LF01]). Das Konzept nutzt einen *Level-of-Information Ansatz*, der 3D Objekte in verteilten Informationsräumen in Form von Repräsentationen mit variablem Gehalt an Information verwendet. Dazu wird die Originalszene in eine Server-Szene und eine Client-seitige Szene aufgeteilt, zwischen denen Objektrepräsentationen ausgetauscht werden können. Es wird zwischen dem Navigationsraum auf Server-Seite und dem Interaktionsraum auf Client-Seite unterschieden, in denen sich jeweils die Objektrepräsentationen befinden. Durch Kombination von Remote 3D Visualisierung im Navigationsraum und von lokaler 3D Visualisierung im Interaktionsraum entsteht der Eindruck einer Gesamtszene während des Arbeitens mit einem 3D Dokument. Dazu werden die Visualisierungsmodule auf Server- und Client-Seite synchronisiert und die Bilddaten vom Server als Textur in die lokale Szene des Clients eingefügt.

Die Verteilung der Visualisierungsprozesse und der Objektrepräsentationen wird durch eine Schieberegler-Funktionalität gesteuert. Der Level-of-Information-Schieberegler (LoI-Regler) kann entweder vom Benutzer selbst eingestellt werden, wobei bestimmte Positionen durch die Randbedingungen gesperrt sein können, oder er wird durch ein spezielles Protokoll automatisch geregelt, wobei die aktuelle Rechtesituation und die Leistung des jeweiligen Clients berücksichtigt werden. Hierdurch kann die *Software-Architektur SCA3D*, die das Konzept der Skalierbaren Szenen realisiert, sich automatisch an die Randbedingungen einer heterogenen Umgebung anpassen. Die An-

passung geschieht durch die Wahl eines entsprechenden Level-of-Information und einer geeigneten Auflösung für jedes bearbeitete Objekt. In Abbildung 5.1 sind die verschiedenen Komponenten des Konzeptes dargestellt. Mehrer Benutzer sind mit einem Visualisierungs-Server verbunden und arbeiten mit einem 3D Dokument, das sich anfangs komplett auf dem Server im Navigationsbereich befindet. Durch Selektion können Objekte in den lokalen Interaktionsraum eines Clients übertragen werden.

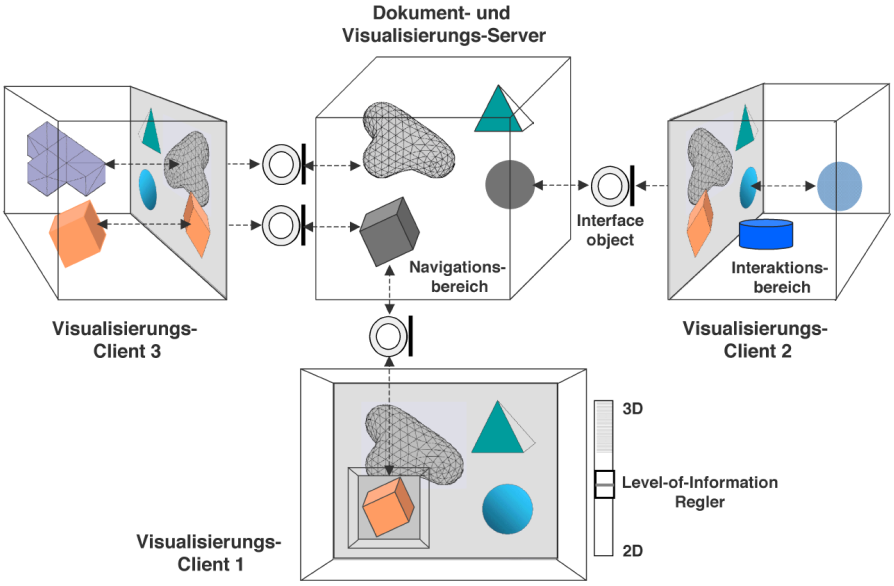

Abbildung 5.1: Das Konzept der Skalierbaren Szenen.

Mit dem Level-of-Information-Schieberegler kann auf Client-Seite kontrolliert werden, welche Objektrepräsentationen im Interaktionsbereich, also auf dem lokalen Rechner des Benutzers, für selektierte Objekte existieren. Ein Objekt kann durch das Original, durch ein polygonales 3D Netz mit einer bestimmten Auflösung, durch Bounding-Box Information oder durch reine Bildinformation zum Client übertragen werden. Je nach Objektrepräsentation wird eine geeignete Rendering-Methode angewendet. Durch diesen Ansatz ergeben sich folgende Vorteile:

- *IPR Management*: Die geistigen Eigentumsrechte für 3D Objekte können geschützt werden, da die Geometrie kontrolliert und stufenweise übertragen wird.
- *Komplexitäts und Lastmanagement*: Durch die Aufteilung in Teilszenen und Verwendung verschiedener Repräsentationen von Objekten kann die Visualisierungslast zwischen Client und Server verteilt werden und die Übertragungsrate reguliert werden.
- *Zusammenarbeit*: Kooperation mehrerer Benutzer ist möglich, wobei mit unterschiedlichen Repräsentationen der 3D Dokumente am Client gearbeitet werden kann. Durch Generierung von entsprechenden Bilddaten am Server können synchronisierte oder jeweils eigene Blickpositionen realisiert werden, so daß entweder alle Benutzer gemeinsam navigieren oder sich selbständig in der Szene bewegen. Dabei werden die Objekte im Interaktionsraum entsprechend synchronisiert.

5.2 Vergleich mit vorhergehenden Arbeiten

In Abschnitt 4.5 des vorhergehenden Kapitel wurden Ansätze zur Kombination von lokalem und Remote 3D Rendering diskutiert und Beispiele in der Literatur beschrieben. In diesem Abschnitt soll eine Einordnung des Ansatzes der Skalierbaren Szenen in den Stand der Technik vorgenommen werden, wobei die jeweils angestrebte Problemlösung eines Ansatzes als Bezugskriterium verwendet wird. Ein Vergleich mit vorhergehenden Arbeiten aus der Literatur soll dabei helfen, die jeweiligen Ziele der bekannten Ansätze zu verdeutlichen.

Die Hauptaspekte der in dieser Arbeit vorgestellten Software-Architektur sind die verteilte 3D Visualisierung von geschützten Dokumenten und die Skalierbarkeit der benötigten Informationsmenge am Client für kooperative Systeme. Dabei wird ein besonderes Augenmerk auf die Anforderungen beim verteilten Arbeiten mit 3D Dokumenten in offenen Informationsräumen, wie digitalen Bibliotheken, gelegt. Diese Anforderungen erfordern eine Berücksichtigung von Sicherheitsaspekten, Netzbandbreiten und lokaler Rechnerleistung. Ein Anwendungsszenario für heutige Systeme, das auch zukünftige Entwicklungen der Informationstechnologie miteinbezieht, kann von einer relativ schnell steigenden Client-Leistung und einer relativ langsamen Verbesserung der Netzanbindung bzw. weiterhin hohen Anbindungskosten in den nächten Jahren ausgehen (siehe z.B. [MCO97]. Die folgende Auflistung der verschiedenen Arbeiten bzw. Anzätze beschreibt die Konzepte anhand der angestrebten Problemlösung, der Randbedingungen bzw. Voraussetzungen und des jeweiligen Anwendungsgebietes. Da in der Literatur bisher nur sehr wenige Systeme beschrieben wurden, die einen kombinierten Ansatz verfolgen, werden hier zwei bekannte Arbeiten mit dem eigenen Ansatz verglichen.

In der Arbeit von Engel et al. [EHTE00] werden lokale und Remote Visualisierungstechniken für **interaktives verteiltes Volumen-Rendering** kombiniert:

- *Problemlösung*: Reduzierung der Client-Last, Ausnutzung von speziellen Hardware- und Software-Komponenten in einer heterogenen Umgebung.
- *Randbedingungen*: Starker, spezialisierter Server-Rechner; schwache oder Standard-PC-Client-Rechner; lokales Netz (LAN: 34 MBit/s TCP/IP); keine Sicherheitsaspekte berücksichtigt.
- *Anwendungsbereich*: Medizinsche Volumendaten, interaktive Visualisierung komplexer Daten im klinischen Einsatz.

Die Arbeit von Mann und Cohen-Or [MCO97] verwendet entfernt gerenderte Texturen, die auf eine einfache, lokale Basisgeometrie abgebildet werden, um Benutzern die **Remote-Navigation in komplexen, virtuellen 3D Welten** auf einem Server zu ermöglichen.

- *Problemlösung*: Reduzierung der Netzlast bei der entfernten Navigation in virtuellen Welten; Reduzierung der lokalen Verzögerung durch Bildextrapolation bei der Navigation.
- *Randbedingungen*: starker Server; schwacher oder Standard-PC-Client; begrentzte Bandbreite der Netzanbindung.
- *Anwendungsbereich*: Remote Navigation in virtuellen Welten im Bereich des World-Wide-Web.

Nun soll im Vergleich das hier untersuchte Konzept der Skalierbaren Szenen anhand der genannten Kriterien eingeordnet werden. Der Ansatz soll, wie oben beschrieben, das **verteilte Arbeiten mit 3D Dokumenten in offenen Informationsräumen** unterstützen:

- *Problemlösung*: Dynamischer Ausgleich von Client- und Server-Last; dynamische Anpassung der Netzlast an die verfügbare Bandbreite, indem die globale Verzögerungszeit laufend ausgewertet wird und der Server mit dementsprechend angepaßten Datenraten sendet; Berücksichtigung von Sicherheitsaspekten; Kooperation in heterogenen Umgebungen.
- *Randbedingungen*: Server-Rechner mit unterschiedlicher Leistung; Client-Rechner mit unterschiedlicher Leistung; variable Anbindungsbandbreiten verschiedener Clients; geschützte Daten.
- *Anwendungsbereich*: Software-Architektur zum verteilten Arbeiten in offenen Informationsräumen; verteilte Visualisierung geschützter 3D Dokumente in Digitalen Bibliotheken; Kooperation mit 3D Dokumenten.

Aus diesem Vergleich der verschiedenen Ansätze geht hervor, daß zum einen die Regulierung der Client-Last bei der 3D Visualisierung mit gleichzeitiger hoher Qualität und Interaktionsrate ein zentrales Ziel bei der Verwendung von kombinierten bzw. hybriden Konzepten ist. Die Leistung von Server-Rechnern spielt ebenso eine wichtige Rolle, da in vielen Fällen Dienste von spezialisierten Hochleistungsrechnem von entfernten Client-Rechnern nutzbar gemacht werden sollen. Dabei ist ebenfalls die Reduzierung der Netzlast in Umgebungen mit niedrigen und begrenzten Bandbreiten von Bedeutung.

Die Herausstellungsmerkmale des Konzeptes der Skalierbaren Szenen in wissenschaftlicher und technischer Hinsicht sind vorallem der dynamische Aspekt, der durch den Level-of-Information Ansatz zusammen mit der Schiebereglerfunktionalität in das Konzept integriert wird, und die enthaltenen Möglichkeiten zum Management von geschützten Daten. Sowohl die dynamischen Anpassung der Client- und Server-Last, als auch die dynamische Regulierbarkeit der Netzlast durch Verwendung von Objektrepräsentationen mit variablem Informationsgehalt sind neue Aspekte in der Entwicklung hybrider Visualisierungsysteme. Die Skalierbarkeit der Informationsmenge am Client-Rechner und die sich daraus ergebenden Vorteile für kooperatives Arbeiten mit 3D Dokumenten sind weitere Punkte, die in der wissenschaftlichen Literatur noch nicht untersucht worden sind.

5.3 Das Konzept der Skalierbaren Szenen in der Gesamtübersicht

In diesem Abschnitt soll ein Überblick des Konzeptes der Skalierbaren Szenen gegeben werden, indem alle wichtigen Komponenten erläutert werden. Das Ziel des Konzeptes ist die Unterstützung des verteilten Arbeitens mit 3D Dokumenten in offenen Informationsräumen, die Informations-Server und Benutzer-Rechner umfassen, durch einen kombinierten Visualisierungsansatz. Zur Visualisierung der 3D Dokumente wird zwischen einem *Navigationsraum* am Server und einem *Interaktionsraum* am Client unterschieden, die über eine Middleware-Schnittstelle synchronisiert werden. Diese Schnittstelle wird in dem vorgestellten Konzept als *Level-of-Information-Raum* (LoI-Raum) bezeichnet, da die Synchronisation über Interface-Objekte stattfindet, die sich in einem virtuellen Raum zwischen Navigationsraum und Interaktionsraum befinden. Zwischen dem Navigationsraum und dem LoI-Raum auf Server-Seite ist ein Filter für Objektrepräsentationen (*IPR-Filter*)

eingefügt, der dafür sorgt, daß nur Objektrepräsentationen zu den Benutzern gelangen, für die sie entsprechende Rechte besitzen. In Abbildung 5.2 ist das Gesamtsystem des Konzeptes der Skalierbaren Szenen gezeigt.

- *Navigationsraum*: Virtueller Raum auf dem Server, in dem sich die server-seitige Szene befindet und worin ein Benutzer durch Remote Visualisierung navigieren kann.
- *Interaktionsraum*: Lokaler 3D Visualisierungsraum der Benutzeranwendung, in dem die Transformationen und die Beleuchtungssimulation der client-seitigen Szene und die lokale Benutzerinteraktion durchgeführt wird.
- *LoI-Raum*: Virtueller Raum am Server, in dem sich die Interface-Objekte für selektierte 3D Objekte befinden. Er ist über eine Middleware-Schnittstelle vom Client aus zugänglich. Durch den IPR-Filter können nur erlaubte Objektformen über den LoI-Raum zum Benutzer gelangen.

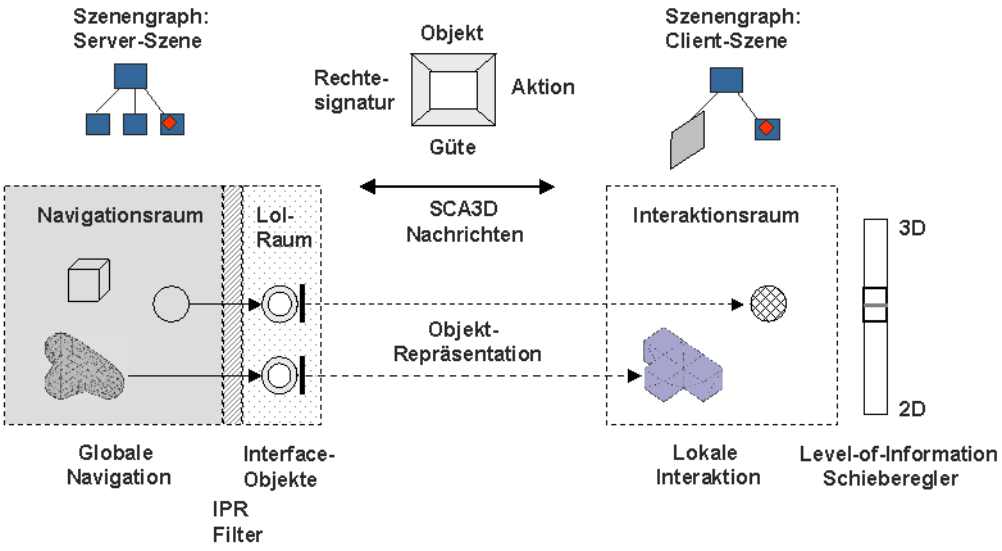

Abbildung 5.2: Gesamtübersicht des Konzeptes der Skalierbaren Szenen

Die Originalszene aus einer Datenbank wird am Server in eine server-seitige Szene und eine client-seitige Szene aufgeteilt. Die client-seitige Szene enthält alle Objekte, die sich im Interaktionsbereich befinden und zusätzlich eine Projektionsfläche für die remote-visualisierten Ansichten der 3D Szene vom Server. Es wird zwischen drei Zuständen für 3D Objekte unterschieden, nämlich *inaktiven Server-Objekten, aktiven Objekten* und *inaktiven Client-Objekten.* In der folgenden Auflistung werden die drei Zustände näher beschrieben:

- *inaktives Server-Objekt*: Wenn ein 3D Dokument am Server in das Visualisierungssystem geladen wird, befinden sich zunächst alle seine Objekte im Navigationsbereich und können

vom Benutzer nur entfernt visualisiert werden. Die Objekte befinden sich für das System dann im Zustand von inaktiven Server-Objekten.

- *aktives Objekt*: Sobald ein Benutzer ein Objekt selektiert, wird ein Interface-Objekt dieses 3D Objektes im LoI-Raum erzeugt, mit dem eine Verbindung zwischen Client und Server entsteht. Das selektierte Objekt befindet sich dann noch auf dem Server, aber gleichzeitig ist am Client eine Objektrepräsentation vorhanden. Das Objekt befindet sich dann im aktiven Zustand und ein Benuzter kann vielfältige Aktionen auf dem Objekt ausführen.

- *inaktives Client-Objekt*: Wenn ein Benuzter die Rechte besitzt, das Original eines serverseitigen Objektes aus dem 3D Dokument zu erwerben und dieses Objekt zum Client-Rechner übertragen worden ist, dann wird das Originalobjekt in der Server-Szene entfernt. Das Objekt, das sich nun lokal beim Client befindet, ist für das System im Zustand eines inaktiven Client-Objektes. Auch Objektrepräsentationen, die nicht dem Original entsprechen, können durch Abkopplung vom Server zu inaktiven Client-Objekten gemacht werden.

In Abbildung 5.2 ist das System mit einem 3D Dokument gezeigt, das gerade bearbeitet wird. Zwei Objekte befinden sich im Zustand von aktiven Objekten und ein Objekt, nämlich der Würfel, ist im Zustand eines inaktiven Server-Objektes. Für die aktiven Objekte existiert ein Interface-Objekt im LoI-Raum und jeweils eine Objektrepräsentation im Interaktionsbereich. Im Fall der Kugel ist dies ein niedrig-aufgelöstes 3D Netz und im Fall des anderen Objektes handelt es sich um ein Subdivision-Surface-Modell, das durch Übertragung von zusätzlichen Parameterwerten schrittweise an die Originalform am Server angeglichen werden kann (*Subdivision Surfaces*, siehe Loop [Loop87] oder Müller und Havemann [MH00]).

Zwischen Client und Server werden während des Arbeitens fortwährend Nachrichten (*SCA3D Nachricht*) mittels eines festgelegten Protokolls ausgetauscht, die den aktuellen Zustand der Objekte und der Visualisierungsanwendungen am Client und am Server im System bekannt machen. Diese Nachrichten enthalten jeweils vier Informationsblöcke, nämlich über das jeweils angefragte *Objekt*, die beabsichtigte *Aktion* auf diesem Objekt, die Leistung bzw. *Güte* eines Teilnehmer-Rechners und schließlich über die aktuelle Rechtesituation zwischen Client und Server in Form einer *Rechtesignatur*. Die Client-Anwendung enthält eine Schiebereglerfunktionalität und ein entsprechendes graphisches Schieberegler-Interface (*Level-of-Information-Schieberegler*), mit dem ein Benutzer wählen kann, welchen Grad des Informationsgehaltes er mit einer Objektrepräsentation übertragen bekommen möchte. Die erlaubten Positionen des Schiebereglers hängen von der Rechtesituation des Benutzers ab. Auch die Leistung eins Client-Rechners und seine Anbindung an den Server werden zur Kontrolle des LoI-Schiebereglers herangezogen, indem ungünstige Positionen bewertet und markiert werden. Die Güte eines Client-Rechners wird mit Hilfe einer Performanzmetrik am Client bestimmt. Die ermittelte Güte wird als skalarer Zahlenwert im System bekannt gemacht und regelmäßig aktualisiert. Im weiteren werden die noch fehlenden Komponenten des Systems aus Abbildung 5.2 beschrieben.

- *SCA3D Nachricht*: Vierseitige Nachrichten, die dem internen Kommunikationsprotokoll des Systems entsprechen, und Auskunft über Objekte, Aktionen, die Güte eines Client-Rechners und die Rechtesituation bei einer Interaktion geben.

- *Level-of-Information-Schieberegler*: Funktionalität der Client-Anwendung, die die Regulierung des Informationsgehaltes von übertragenen Objekten erlaubt. Die Positionen des LoI-Schiebereglers geben an, in welcher Form ein Objekt in den Interaktionsraum übertragen wird.

- *Performanzmetrik*: Im System bekannte Metrik, die einen Vergleich der Leistung und Anbindung von verschiedenen Client-Rechnern ermöglicht. Die Güte wird am Client durch Messungen und anhand der Performanzmetrik bestimmt.

Die Freigabe und Bewertung von Positionen des Schiebereglers werden im System automatisch vorgenommen, so daß die Benutzerwahl jederzeit den Randbedingungen der Umgebung angepaßt ist. Bei einer Vollautomatisierung der Kontrolle wird jeweils die optimale Reglerposition vom System für einen Benutzer gewählt. Dies wird im Abschnitt 5.8 besprochen und erlaubt ein adaptives Verhalten des Systems durch dynamische Kontrolle des Schiebereglers. In den folgenden Abschnitten werden die einzelnen Komponenten und ihre Modellierung näher beschrieben.

5.4 Der Level-of-Information Ansatz

In diesem Abschnitt wird der Level-of-Information Ansatz, der zentrale Bedeutung für das Konzept der Skalierbaren Szenen hat, vorgestellt und näher erläutert. Der Ansatz beruht auf der Grundidee, daß in einer verteilten visuellen Mehrbenutzerumgebung eine Möglichkeit zur Anpassung der Informationsmenge für jeden Client an seine lokalen Gegebenheiten und Anforderungen bestehen sollte. Sowohl die lokale Leistung und Anbindung, als auch der vom Benutzer aktuell verlangte Detaillierungsgrad der Information sollen von einem verteilten System berücksichtigt werden, um das Arbeiten in heterogenen Mehrbenutzerumgebungen optimal zu unterstützen. Der Level-of-Information Ansatz setzt diese Idee um, indem für 3D Dokumente und ihre Objekte verschiedene Repräsentationen bzw. Formen für das verteilte Arbeiten angeboten werden. Wie in Kapitel 3, Abschnitt 3.1 schon angesprochen, kann ein 3D Objekt auf verschiedene Weise beschrieben werden. Für geometrische Objekte, die implizit durch Angabe von Operationen und Parameter modelliert werden, existieren eine Anzahl von Repräsentationsformen, die alle das gleiche Objekt darstellen, aber einen unterschiedlichen Gehalt an Information über die Geometrie enthalten. Die Skala der Beschreibungsarten von 3D Objekten reicht von der exakten parametrisierten 3D Form, über polygonale 3D Netze bis hin zu reiner Bildinformation. Für das Arbeiten mit diesen Objekten in Visualisierungsumgebungen sind alle diese Formen für bestimmte Aufgabenbereiche geeignet. Daher liegt es nahe, alle Objektformen auf dieser Skala in einer verteilten Software-Architektur zu unterstützen und damit ein hohes Maß an Flexibilität beim Arbeiten mit 3D Dokumenten zu ermöglichen.

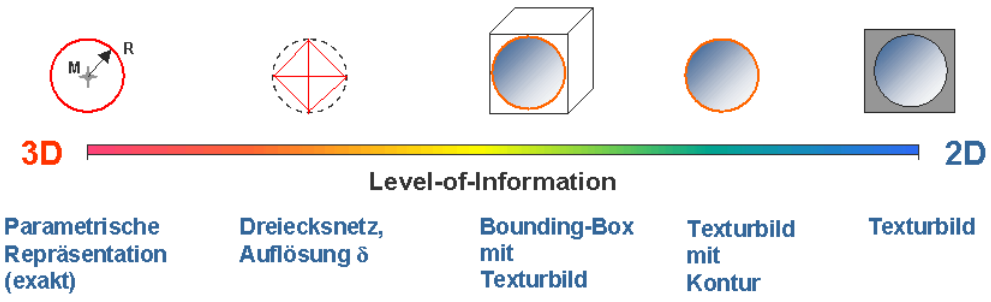

Abbildung 5.3: Objektrepräsentationen von 3D Objekten mit variablem Gehalt an Information.

Aus den vorher genannten Gründen und Anforderungen, die die lokale Rechnerleistung, Netzanbindung oder Sicherheitsaspekte von Originaldaten umfassen, bietet die Interaktion mit den Originalobjekten über Objektformen mit geringerem Informationsgehalt oft Vorteile. Dies wird vom Level-of-Information Ansatz ermöglicht, indem die Originaldaten eines 3D Objektes auf dem Server bleiben und der Benutzer für einzelne Objekte verschiedene Objektrepräsentationen anfordern kann, die dann zu ihm übertragen werden. Solange die Objektrepräsentation in einem aktiven Zustand ist, bleibt sie dabei mit ihrem Original am Server verknüpft. In Abbildung 5.3 ist die Skala der Objektformen mit einem variablem Gehalt an Information gezeigt, die von der exakten parametrisierten Repräsentation bis zur reinen Bilddatenrepräsentation in Form einer Textur reicht. Für jeden Level existieren verschiedene Auflösungsgrade, die z.B. durch die Maschengröße bei Dreiecksnetzen oder die Bildauflösung bei Bildobjekten bestimmt werden.

Aufbau einer Client-Szene

Durch den Zugriff auf Originalobjekte am Server über verschiedene Objektformen können die Anforderungen verteilter Umgebungen flexibler erfüllt werden. Ein Benutzer, der lediglich einen relativ schwachen Rechner mit einer langsamen Anbindung, z.B. über ein Modem, besitzt, wird nicht eine vollständige Kopie des komplexen 3D Dokumentes, mit dem er gerade arbeitet, auf seinen lokalen Rechner übertragen wollen. Wenn er nun mit einem Teildokument arbeitet, das er in niedriger Auflösung benötigt, wird er eine entsprechende Kopie dieses Teildokuments anfordern, die er mit seinen lokalen Randbedingungen komfortabel bearbeiten kann. Die übrige Information des komplexen Dokumentes wird über eine regelmäßigen Aktualisierung eines Hintergrundbildes mittels Remote-Visualisierung zugänglich gemacht. Zu diesem Zweck enthält die Client-Szene eine Projektionsfläche für Texturen, die vom Server geliefert werden. In Abbildung 5.4 ist die Client-Szene eines Benutzers schematisch mit ihrem jeweiligen Szenengraphen gezeigt.

Am Beginn einer Visualisierungssitzung enthält die Client-Szene nur die 2D Projektionsfläche für die Remote-Visualisierung des server-seitigen 3D Dokumentes (Abbildung 5.4: linke Seite). Hier ist das 3D Dokument als einzelnes Objekt in Form eines Bildobjektes am Client zugänglich. Der Benutzer kann nun durch Selektion von Einzelobjekten oder Objektgruppen auf seinem Display Objekte in verschiedenen Formen anfordern, die dann vom System in seine lokale Szene integriert werden. In der Mitte von Abbildung 5.4 sind zwei Objekte aus dem Bildhintergrund herausgelöst und in die Client-Szene integriert worden. Eine zusätzliche lokale Lichtquelle sorgt dafür, daß die lokale Visualisierung sinnvolle Ergebnisse liefert, falls keine Lichtquellen mitübertragen wurden. Wenn der Benutzer nun in der Gesamtszene navigiert, werden der Bildhintergrund und die Vordergrundobjekte synchronisiert, sodaß ein visueller Gesamteindruck entsteht. Sobald der Benutzer das gesamte 3D Dokument als Kopie des Originals auf seinen Rechner übertragen hat, wird keine Hintergrundinformation vom Server mehr benötigt und die gesamte Visualisierung geschieht lokal (Abbildung 5.4: rechte Seite).

Die Übertragung von Objektrepräsentationen wird, wie oben angedeutet, vom Benutzer initiiert und vom Server durch eine Push-Operation ausgeführt. Dadurch wird der Zugriff auf unerlaubte Objektdaten unmöglich gemacht. Der LoI-Schieberegler steuert hierbei den Informationsgehalt der übertragenen Objektform.

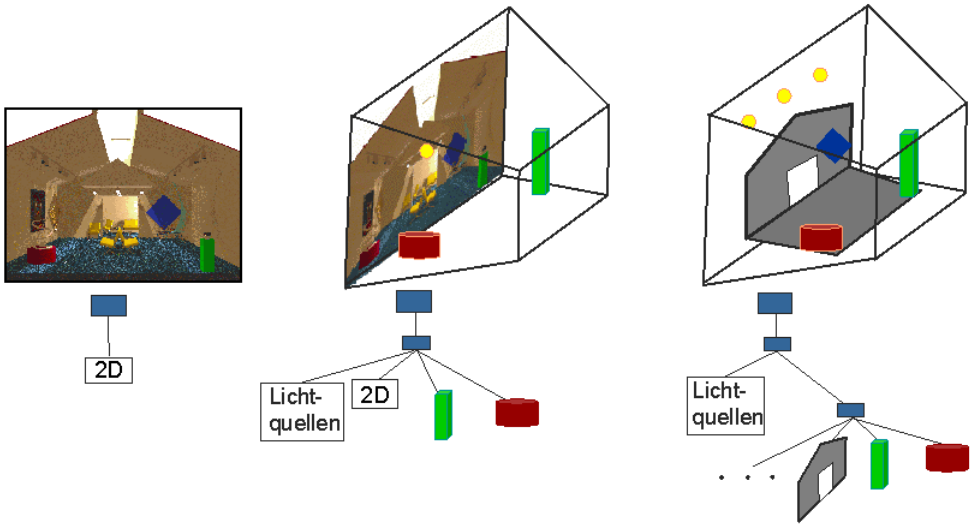

Abbildung 5.4: Repräsentation eines 3D Dokumentes auf Client-Seite als Bildobjekt (links), als Kombination aus 3D- und Bildobjekten und als 3D Gesamtszene.

5.5 Modellierung von Navigations- und Interaktionsraum mit einem Document-Request-Broker-Ansatz

In diesem Abschnitt soll die Modellierung der im Konzept der Skalierbaren Szenen enthaltenen Navigations- und Interaktionsräume vorgestellt werden. In Abschnitt 5.3 wurde erläutert, daß für aktive 3D Objekte ein Interface-Objekt erzeugt wird, das sich in einem virtuellen Level-of-Information Raum befindet. Diese Interface-Objekte dienen dazu, dem Benutzer den indirekten Zugriff auf die geschützten server-seitigen 3D Objekte zu ermöglichen. Neben der Erzeugung von neuen Objektrepräsentationen wird auch die Manipulation und Transformation von 3D Objekten auf dem Server mit Hilfe der Interface-Objekte realisiert. Die Interface-Objekte stellen Methoden für den Zugriff auf Objektdaten zur Verfügung, auf die eine Client-Anwendung über ein Referenzobjekt zugreifen kann. Die Referenzobjekte werden im Document-Request-Broker Ansatz durch die Objektrepräsentationen der aktiven 3D Objekte zur Verfügung gestellt.

Das in Kapitel 2 vorgestellte Szenengraphkonzept für 3D Dokumente, das von vielen modernen Visualisierungsanwendungen unterstützt wird, legt es nahe, verteilte Interaktion mit 3D Dokumenten über eine allgemeine Vermittlungsschicht umzusetzen, die zwischen der Anwendungs- und Middleware-Schicht liegt. Diese Vermittlungsschicht, die im folgenden als *Document-Request-Broker* bezeichnet wird, arbeitet mit Interface-Objekten auf Server-Seite bzw. Objektreferenzen auf Client-Seite und ähnelt damit dem Aufbau einer objekt-orientierten Middleware-Architektur, wie z.B. CORBA.

In Abbildung 5.5 ist die Systemmodellierung mittels einer Vermittlunsgschicht für verteilte Objektanfragen, dem Document-Request-Broker, dargestellt. Auf unterster Ebene laufen alle objektorientierten Methodenaufrufe über die verwendete Middelware. Oberhalb dieser Middleware-

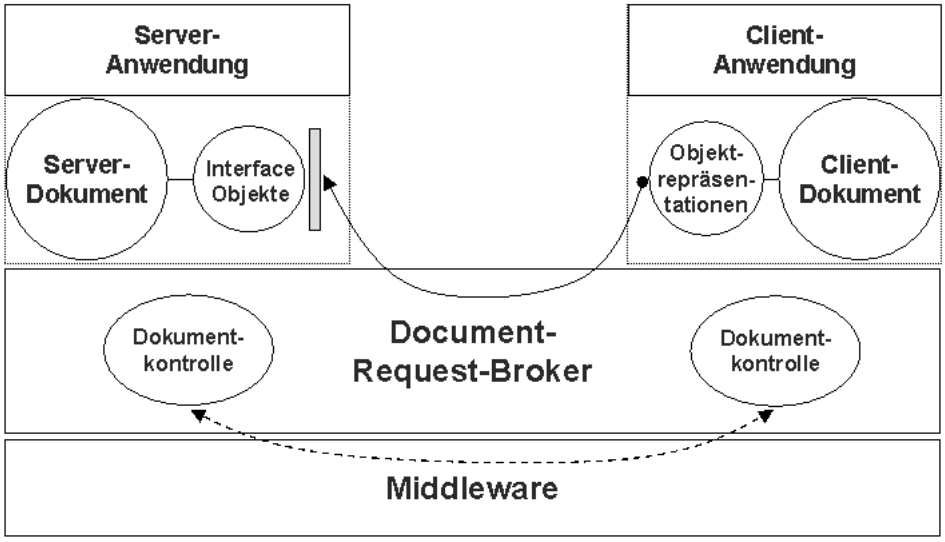

Abbildung 5.5: Konzept einer Document-Request-Broker Architektur für objekt-orientierte Dokumentmodelle.

Schicht ist nun aber eine Schicht zur Kontrolle des Dokumentes auf dem Server und des dazugehörigen Dokumentes auf Client-Seite eingefügt. Alle Operationen die vom Benutzer auf das bearbeitete Dokument angewendet werden, werden mittels der Dokumentkontrolle auf Server-Seite und Client-Seite synchronisiert. Dadurch wird für aktive Objekte eine Verknüpfung zwischen Client und Server aufrechterhalten. Auf diese Weise wird auch die Synchronisation der Benutzernavigation im Gesamtsystem ermöglicht.

Auf der Anwendungsebene werden für alle Objekte, die vom Benutzer durch Selektion in den aktiven Zustand gebracht wurden, Interface-Objekte erzeugt. Mit der Erzeugung wird gleichzeitig eine Objektrepräsentation als Referenzobjekt zum Client übertragen. Die Aktionen, die ein Benutzer nun an seiner lokalen Objektform vornimmt, werden über den Document-Request-Broker zum server-seitigen Interface-Objekt übertragen und dann auf den Szenengraph des Originaldokumentes angewendet. Dies ist in Abbildung 5.5 durch einen Pfeil zwischen Objektrepräsentation und Interface-Objekt verdeutlicht.

Am Beginn einer Visualisierunsgsitzung steht dem Benutzer lediglich eine Objektrepräsentation des server-seitigen Gesamtdokumentes als Bildobjekt zur Verfügung. Durch Navigations- und Selektionsbefehle kann er sich in der Szene bewegen bzw. neue Objektrepräsentationen anfordern. Diese Benutzeraktionen werden stets über die Vermittlungsschicht zum Server-Dokument übertragen. Das Basis-Interface-Objekt der Gesamtszene stellt Methoden für die Navigation und Selektion von Objekten zur Verfügung. Der Entwurf dieses Ansatzes mit Hilfe des objektorientierten Dokumentmodells einer szenengraph-basierten Anwendungsschicht (OpenInventor) und einer Middleware-Schicht (CORBA) wird in Kapitel 6 im Detail beschrieben. Die zentrale Bedeutung des Ansatzes für die Gesamtarchitektur sei jedoch schon hier betont.

5.6 Management von Sicherheitsinformationen: Ein IPR-Filter

Um den Anforderungen durch Sicherheitsaspekte in verteilten Umgebungen gerecht zu werden, muß das Konzept der Skalierbaren Szenen Möglichkeiten zur Integration, Extraktion und Evaluation von sicherheitsrelevanten Informationen bieten. Hier soll es darum gehen, die Möglichkeiten zum Management von geschützten Dokumenten und den entsprechenden Sicherheitsinformationen im Konzept der Skalierbaren Szenen zu erläutern. Anhand des Ablaufs der relevanten Prozesse beim Arbeiten mit geschützten 3D Dokumenten innnerhalb der Software-Architektur SCA3D soll das integrierte Sicherheitsmanagement verdeutlicht werden. In Abbildung 5.6 ist dieser Ablauf beim Zugriff und bei der Selektion eines 3D Dokumentes durch einen Benutzer dargestellt.

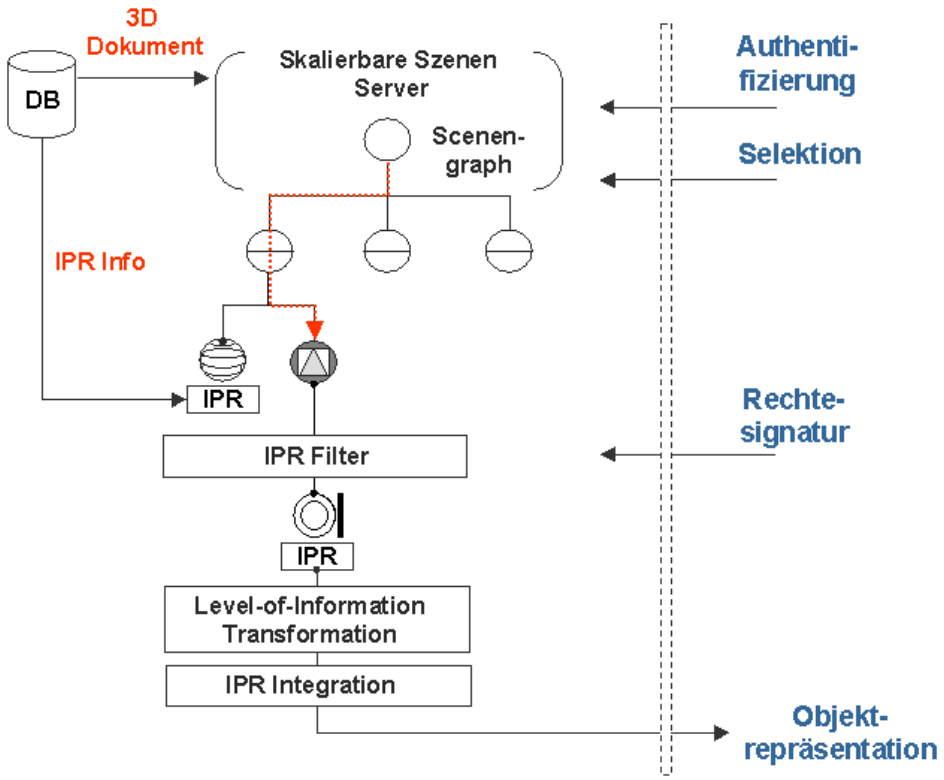

Abbildung 5.6: Sicherheitsmanagement im Konzept der Skalierbaren Szenen.

In Abschnitt 2.4 sind als die relevanten Schritte eines umfassenden Sicherheitsmanagements die Authentifizierung, die Rechteverhandlung, die Transformation von Objekten (z.B. durch Integration von IPR-Daten) und die Verschlüsselung bei der Übertragung erkannt worden. An dieser Stelle wird nun auf die Umsetzung der ersten drei Schritte, nämlich Authentifizierung, Rechteverhandlung und Objekttransformation, im Konzept der Skalierbaren Szenen eingegangen. Der letzte Schritt, die Veschlüsselung, soll hier nicht weiter beschrieben werden, da er nicht integraler Teil des Konzeptes ist und durch Verwendung von bekannten Verschlüsselungstechniken eingefügt werden kann. Im

folgenden werden die genannten Schritte näher ausgeführt. Die in der Abbildung zentralen Prozesse und Daten werden in der Auflistung an entsprechender Stelle hervorgehoben :

- *Authentifizierung*: Die **Authentifizierung** eines Benutzers geschieht beim Anmelden am System-Server und führt zu einer Einordnung des Benutzers in eine Kategorie, wie *Gast, eingetragenes Mitglied* oder *Administrator*. Jeder Kategorie entspricht ein Rechtezustand, der in Form einer Signatur an den Benutzer zurückgesendet wird. Jede Client-Anwendung erhält beim Anmelden außerdem eine eindeutige Kennung, die mit der Signatur übertragen wird. Mitglieder und Administratoren besitzen vor der Anmeldung bereits eine Signatur, die sie durch eine vorhergehende Operation erhalten haben. Durch Abgleich dieser Signatur mit der am Server für diesen Benutzer gespeicherten Signatur wird eine Authentifizierung gewährleistet. Gastbenutzer erhalten minimale Rechte und eine Signatur, die am Ende einer Sitzung verfällt.

- *Rechteverhandlung*: Nach der erfolgreichen Anmeldung wird vom Benutzer ein Dokument aus einer Datenbank in den System-Server durch Retrieval geladen. Gleichzeitig werden sicherheitsrelevante Informationen, wie z.B. Daten über Copyright, ebenfalls aus einer Datenbank in das interne Dokumentmodell eingefügt, das im Fall von **3D Dokumenten** dem Szenengraph entspricht. Im hier beschrieben Konzept wird diese **IPR-Information** in Form von Attributknoten in die Untergruppen der betreffenden Inhaltsobjekte eingefügt. Bei einem Zugriff auf einzelne Inhaltsobjekte des Dokumentes durch **Selektion** wird ein Interface-Objekt erzeugt, das die zugehörigen IPR-Daten in Zustandsfeldern enthält. Mit Hilfe der gleichzeitig mit der Benutzeraktion gesendeten **Rechtesignatur** wird nun die Methodenschnittstelle des Interface-Objektes an die aktuelle Rechtesituation angepasst. Dies geschieht durch Öffnung bzw. Sperrung der benötigten Abläufe im Interface-Objekt, wobei die in der Signatur enthaltenen LoI-Daten verwendet werden. Wenn ein Benutzer zusätzliche Rechte erwirbt, wird die Methodenschnittstelle entsprechend erweitert. Diese Anpassung der Methodenschnittstelle des Interface-Objektes entspricht der IPR-Filter-Funktionalität in Abbildung 5.6.

- *Objekttransformation*: Sobald ein Benutzer ein 3D Objekt selektiert hat, wird eine **Objektrepräsentation** in seine Client-Szene eingefügt. Dies geschieht durch die Erzeugung einer den Rechten entsprechenden Objektform bei der **Level-of-Information-Transformation**. Diese Objektrepräsentation wird durch geeignete Methoden des Interface-Objektes aus den Originaldaten generiert, z.B. wird aus einem parametrisierten Oberflächenobjekt eine polygonales 3D Netz erzeugt. Anschließend wird in diese Objektrepräsentation eine Kopie der IPR-Daten integriert (**IPR Integration**), indem eine verschlüsselte Signatur daran angehängt oder ein digitales Wasserzeichen darin eingebettet wird. In dieser Form wird eine Kopie der originalen Objektdaten zum Client-Rechner übertragen.

Bei jeder weiteren Interaktion mit einem Interface-Objekt wird die Rechtesignatur mit den IPR-Daten des Dokumentes verglichen, um die aktuelle Rechtesituation festzustellen. Nur wenn eine Aktion erlaubt ist, wird sie vom System auch ausgeführt. Durch dieses Konzept der Rechteverwaltung wird es möglich, mit geschützten 3D Dokumenten in offenen Informationsräumen flexibel zu arbeiten.

5.7 Das Kooperationsmodell der Skalierbaren Szenen

Das Konzept der Skalierbaren Szenen soll die Kooperation zwischen mehreren Benutzern in offenen Informationsräumen unterstützen. Zu diesem Zweck wird ein Kooperationsmodell in das System integriert, das im folgenden vorgestellt wird.

Eine Software-Architektur, die das Konzept der Skalierbaren Szenen umsetzt, stellt mit jeder Server-Anwendung und mit jeder Client-Anwendung einen Methodenkern zur Verfügung, der zur Navigation in und zur Interaktion mit dem Gesamtdokument dient. Dieser Kern von Methoden ist direkt mit der server-seitigen und client-seitigen Szene verbunden und wird als Basis für die Interaktion über den Document-Request-Broker verwendet. Die Methoden selbst sind Teil der Dokumentkontroll-Objekte aus Abbildung 5.5 und werden über RPCs mit Hilfe der unterliegenden Middleware-Schicht aufgerufen. In Abbildung 5.7 sind ein Server und zwei Client-Anwendungen zusammen mit ihrem Methodenkern dargestellt. Das Symbol des Kreises mit einem eingeschriebenen Kreuz stellt dar, daß diese Methode über einen RPC in der verteilten Umgebung aufgerufen werden kann. Die Namen der Objektreferenzen sind hier wegen der Nähe zum CORBA-Standard in der englischen Sprache benannt. Die wichtigsten Methoden des Kerns werden im folgenden beschrieben:

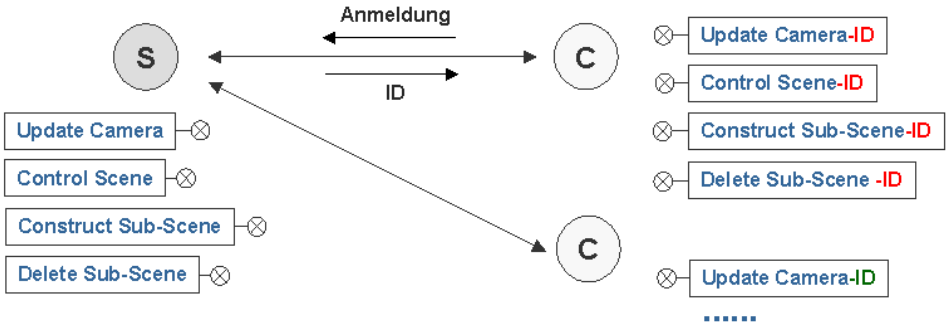

Abbildung 5.7: Das Kooperationsmodell der Skalierbaren Szenen mit dem Methodenkern.

- *Update Camera*: Mit dieser Methode werden Kameraparameter eines Teilnehmers auf den Szenengraph eines anderen Teilnehmers übertragen. Dadurch ist eine synchronisierte Navigation zwischen den beiden Benutzern möglich.
- *Control Scene*: Hiermit wird die Menge der Methoden bezeichnet, die zur Interaktion auf Objektebene nötig sind. Im einzelnen werden Transformationen, wie z.B. Verschiebung oder Skalierung, und lokale Manipulationen, wie z.B. Änderung von Eckpunktkoordinaten, mit Hilfe dieser Methoden ausgeführt. Konzeptionell sind diese Methoden Teil der Methodenschnittstelle der Interface-Objekte und werden dort angegeben. Als Ausgangsbasis werden Methoden zur Selektion von Objekten bereitgestellt.
- *Construct Sub-Scene*: Diese Methode dient zum Einfügen von Teildokumenten bzw. Objekt-

gruppen in einen bestehenden Szenengraph. Objektrepräsentationen werden auf diese Weise in die Client-Szene eingefügt.

- *Delete Sub-Scene*: Mit dieser Methode werden Teildokumenten bzw. Objektgruppen aus einem Dokument entfernt. Dies ist unter anderem nötig, wenn ein Objekt als Originalkopie zum Client übertragen wird und das Original aus der Server-Szene gelöscht wird.

Da jeder Teilnehmer im System, also die Server-Anwendung und alle Client-Anwendungen, diesen Methodenkern zum Zugriff auf ihr lokales Gesamtdokument bereitstellen, müssen die Objektreferenzen der Middleware-Schnittstelle für jeden Teilnehmer unterschiedlich benannt werden. Dies wird hier realisiert, indem an die Namen der Objektreferenzen der Client-Methoden die Client-Kennung (**ID**) angehängt wird und diese Namen dann im System bekannt gemacht werden. Dies ist möglich, da jeder Client-Anwendung bei der Anmeldung eine eindeutige Kennung zugewiesen wird. Die Referenznamen für den Methodenkern des Servers sind nicht mit einer zusätzlichen Kennung versehen, so daß sie von jeder Client-Anwendung sofort aufgerufen werden können. Durch diese Namensvergabe kann jede Client-Anwendung vom Server individuell angesprochen werden, so daß das System sehr flexibel im Hinblick auf verschiedene Kooperationsszenarien wird.

5.8 Adaptives Verhalten durch dynamische Kontrolle des LoI-Reglers

Beim Arbeiten mit komplexen Dokumenten in verteilten Umgebungen ist es wichtig, daß die zugrundeliegende Software-Architektur Möglichkeiten zur Anpassung der Belastung von Client-Ressourcen und des Netzwerkes bietet. In diesem Abschnitt wird beschrieben, welche Mechanismen das Konzept der Skalierbaren Szenen und damit die Software-Architektur SCA3D für diese Adaption bietet.

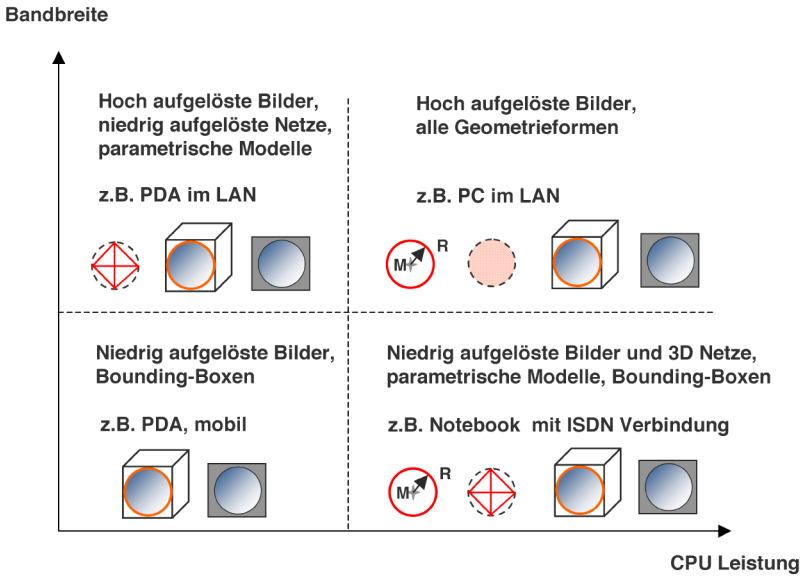

Abbildung 5.8: Klassifizierung von Client-Anwendungen und Zuordnung von Objektrepräsentationen mit geeignetem Level-of-Information.

Die grundlegende Idee ist es, die Informationsmenge, die übers Netz transportiert wird und am Client verarbeitet werden muß, mit Hilfe des Level-of-Information-Schiebereglers zu kontrollieren. In Abbildung 5.8 ist eine Klassifizierung von Client-Anwendungen bezüglich der Leistung und Anbindung der genutzten Rechner gezeigt. Das Spektrum reicht von mobilen Endgeräten (Mobiler PDA), über leistungsschwache Geräte mit guter Anbindung (PDA im LAN), über leistungsstarke Rechner mit schlechter Anbindung (PC über Modem) bis hin zu leistungsstarken Rechnern mit sehr guter Anbindung (PC im LAN). Je nach Leistung und Anbindung wird das Verhältnis zwischen lokaler und Remote-Visualisierungslast angepaßt. Dies geschieht durch Auswahl und Verteilung geeigneter Objektrepräsentationen, die mit den Symbolen aus Abschnitt 5.4 ebenfalls in der Abbildung gezeigt sind. Die Position des LoI-Schieberegler sorgt dafür, daß jeweils die Objektrepräsentationen an die Benutzer verteilt werden, die den Randbedingungen am besten entsprechen.

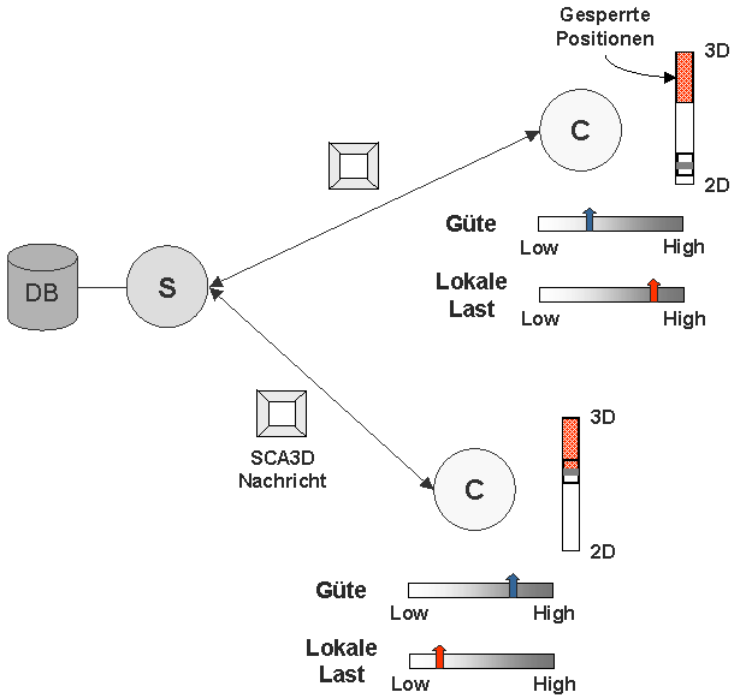

Abbildung 5.9: Dynamische Kontrolle der Schiebereglerpositionen in einer verteilten Mehrbenutzerumgebung.

In Abbildung 5.9 ist ein System mit einem Informations-Server und zwei angebundenen Client-Rechnern gezeigt. Die Benutzer arbeiten gemeinsam mit einem 3D Dokument, das in den Anwendungs-Server geladen ist. Die verfügbare Bandbreite und die lokale CPU-Last können variieren, da sich die Gesamtnetzlast verändern kann bzw. zusätzliche Prozesse auf einer CPU ablaufen können. Jede Client-Anwendung stellt einen LoI-Schieberegler zur Verfügung, mit dem die Repräsentation und Auflösung des zu selektierenden Objektes beeinflusst werden kann. Am Client wird die aktuelle Last laufend ausgewertet, indem die lokale Bildwiederholrate der Client-Anwendung unter festgelegten Bedingungen gemessen wird. Die Güte eines Client-Rechners wird bestimmt, indem regelmäßig das optimale Verhältnis von lokaler und Remote-Framerate und des-

sen Änderung bei variierender Informationsmenge am Client ermittelt wird. Die Güte gibt Auskunft darüber, wie groß die Leistung eines Clienten aktuell ist und wie sich diese unter variierenden Bedingungen im System vorraussichtlich ändern wird.

Die Einstellung des LoI-Reglers wird einerseits durch die Rechtesituation zwischen Server und Client bestimmt und andererseits durch die Gütewerte des Client-Rechners. Die Rechtesituation führt dazu, daß Bereiche des Schiebereglers ganz gesperrt sein können. Dies ist in Abbildung 5.9 durch eine rote Schraffierung dieser Positionen des LoI-Reglers verdeutlicht. Durch Auswertung der Gütewerte wird die optimale Position des LoI-Reglers einer Client-Anwendung für die nächsten Interaktionsschritte festgelegt. In Abbildung 5.9 hat die untere der beiden Client-Anwendungen einen hohen Gütewert und eine aktuell niedrige Last. Daher steht der LoI-Regler auf der Position, die eine Übertragung von möglichst viel geometrischer Information erlaubt. Die Positionen oberhalb sind durch die Rechtesituation gesperrt. Der andere Benutzer arbeitet mit einem Rechner, der eine geringe Güte und bereits eine hohe Last aufweist.

5.8.1 Leistungsmessung und Regelgrößen

Meßgrößen im Gesamtsystem sind die lokale Framerate (LFR) und die Remote-Framerate (RFR), die als Kehrwert der Verzögerungszeit zwischen einer Client-Aktion und des visuellen Feedbacks vom Server ermittelt wird. Die lokalen Verzögerung wird als annähernd gleich zum Kehrwert der lokalen Framerate angenommen. Weitere Meßgrößen sind die aktuelle Informationsmenge am Server DS_S (*Document Size*) und am Client DS_C, ausgedrückt durch die Anzahl der gerenderten Dreiecke einer Szene. Weiterhin werden die insgesamt vom Server ausgegebene Datenrate DF_S (*Data Flow*) und die am Client eingelesene Datenrate DF_C fortlaufend bestimmt.

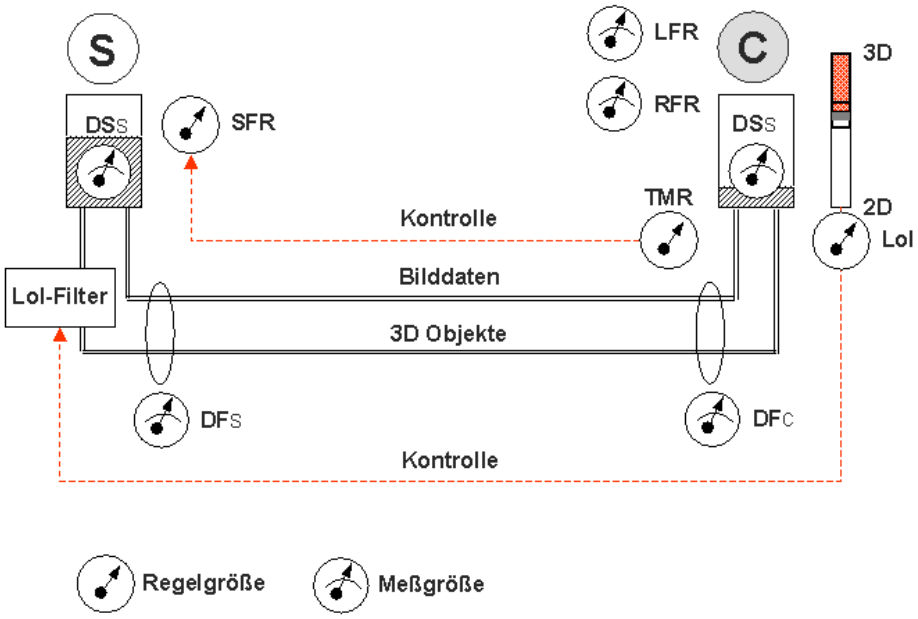

Abbildung 5.10: Regel- und Meßgrößen in der SCA3D Architektur.

- *Lokale Framerate*: Die lokale Framerate *LFR* gibt die Bildwiederholrate der Visualisierungsanwendung des Benutzers an. Die Leistung des Client-Rechners wird als direkt proportional zur lokalen Framerate angesehen und ist näherungsweise gleich dem Kehrwert der lokalen Verzögerung. Dies ist in Formel 5.1 angegeben.

$$LFR \tilde{=} \frac{1}{T_{Lokal}} \tag{5.1}$$

- *Remote-Framerate*: In die Remote-Framerate *RFR* geht nicht nur die Server-Framerate *SFR* mit ein, sondern sie wird auch von der Übertragungsverzögerung und damit von der Netzanbindung beeinflußt, da sie aus dem Kehrwert der Gesamtverzögerungszeit des visuellen Feedbacks vom Server berechnet wird. Diese Gesamtverzögerungszeit setzt sich aus der Zeit zur Erzeugung eines aktuellen Bildes des veränderten Modells T_{Remote}, der Zeit zur Hinübertragung der Benutzeraktion und Rückübertragung der Bilddaten T_{Net} und schließlich der Zeit zusammen, die benötigt wird, um das Bild als Textur am Client anzuzeigen $T_{Texture}$. Dies ist in Formel 5.2 wiedergegeben, in der die Remote-Framerate als Kehrwert der Gesamtverzögerungszeit angegeben ist. Die lokale Rate, mit der Texturen von der lokalen Visualisierungsanwendung angezeigt werden, wird als (*Texture-Mapping-Rate*: TMR) bezeichnet.

$$RFR \tilde{=} \frac{1}{T_{Remote} + T_{Net} + T_{Texture}} \tag{5.2}$$

  Die Remote-Framerate wird im Gesamtsystem gesteuert, indem die Texture-Mapping-Rate variiert und die server-seitige Bilderzeugungsrate automatisch daran angepaßt wird. Die Messung der Gesamtverzögerungszeit geschieht, indem beim Auslösen einer Benutzeraktion eine Kennung zusammen mit einer Startzeit am Client erzeugt und mit den Aktionsdaten zum Server gesendet wird. Dort wird, nachdem die Aktion auf das Modell übertragen wurde, in das nächste generierte Bild die Aktionskennung und die Kennung der Client-Anwendung in einen festen Pixelbereich binär eingebracht. Am Client wird jeweils dieser bekannte Pixelbereich darauf untersucht, ob die aktuelle Aktionskennung des Clients dort eingebracht wurde. Wenn dies zutrifft, wird eine Endzeit ermittelt und die Gesamtzeit der Aktion ausgewertet.

- *Informationsmenge am Server und Client*: Bei der Visualisierung wird sowohl am Server, als auch am Client die aktuelle Größe des 3D Dokumentes DS_S bzw. DS_C laufend bestimmt. Sie werden aus der Anzahl der zu berechnenden Dreiecke ermittelt.

- *Datenrate am Server und Client* : Vom Server werden sowohl Bilddaten, als auch Objektdaten auf das Netzwerk geschickt, die entweder über Socket-Verbindungen (Bilddaten) oder mittels des CORBA-interne Datenprotokolls (Objektdaten) zum Client übertragen werden. Durch Protokollierung der Datenmenge pro Zeiteinheit werden die Gesamtdatenraten von Server und Client DF_S bzw. DF_C gemessen. So können indirekt Aussagen über den Datenfluß und über die Anbindung getroffen werden.

In einem System, das das Konzept der Skalierbaren Szenen umsetzt, existieren variable Parameter, die als Regelgrößen bezeichnet werden. Die entscheidenden Regelgrößen in der SCA3D Architektur sind einerseits der Level-of-Information-Parameter, der den Betrag der mit einem Objekt übertragenen Information regelt, und andererseits die Texture-Mapping-Rate, die angibt, wie

oft die Textur auf der Projektionsfläche am Client aktualisiert wird. In Abbildung 5.10 ist ein solches System mit den beiden Regelgrößen und den oben beschriebenen Meßgrößen gezeigt. Zusätzlich wird die server-seitige Framerate *SFR* als Regelgröße betrachtet, die jedoch an die Kontrolle der Texture-Mapping-Rate am Client gekoppelt ist. Durch Beinflussung der client-seitigen Texture-Mapping-Rate kann daher die vom Server gelieferte Bilddatenmenge gesteuert werden. Gleichzeitig wird davon auch die lokale Framerate beeinflußt. Es handelt sich hierbei um einen Regelkreis zwischen Client und Server, der so eingestellt wird, daß für einen Benutzer das optimale Verhältnis zwischen lokaler und Remote-Framerate gefunden wird.

Die gemessenen Werte werden lokal verarbeitet oder über die Middleware-Schnittstelle im System bekannt gemacht. Die Auwertung der Meßwerte geschieht laufend und die Resultate werden in Form einer Client-Güte ausgedrückt, der die im folgenden beschriebene Performanzmetrik zugrundeliegt.

5.8.2 Festlegung einer Metrik zur Leistungsmessung

Die Bestimmung des optimalen Verhältnisses zwischen lokaler und Remote-Framerate bei einer gegebenen Szenengröße geschieht durch eine schrittweise Erhöhung der Texture-Mapping-Rate von einem minimalen Wert bis zu ihren maximalen Wert, auf den das System noch reagiert. Dieser Auswertungsschritt wird in regelmäßigen Abständen wiederholt. Aus den entstehenden Meßreihen, die den Verlauf der lokalen und der Remote-Framerate in Abhängigkeit von der Texture-Mapping-Rate wiedergibt, kann das optimale Verhältnis der lokalen und Remote-Framerate bestimmt werden. Dies ist in Abbildung 5.11 gezeigt. Der optimale Punkt ist hier durch die größte Steigung der Kurve definiert.

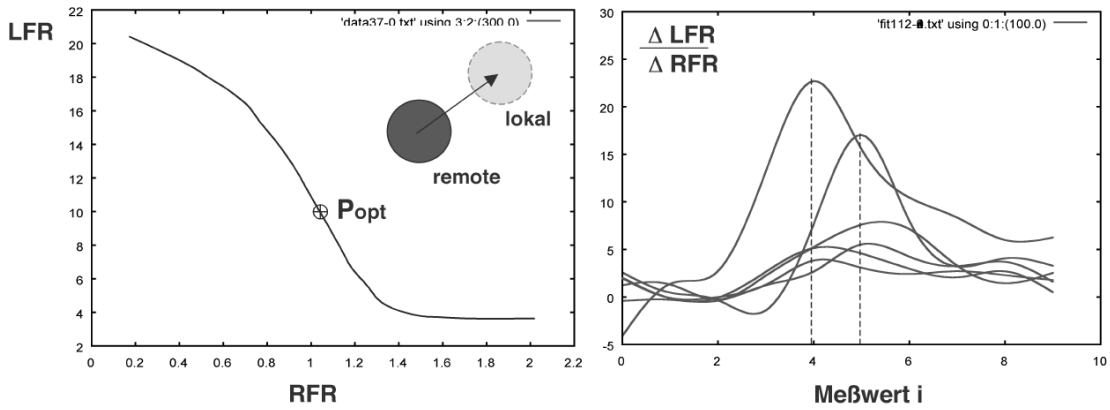

Abbildung 5.11: Bestimmung der optimalen Frameraten des Clients und des Servers.

Auf der linken Seite von Abbildung 5.11 ist der Verlauf der LFR-RFR-Kurve gezeigt, der mit dem Wert der Texture-Mapping-Rate parametrisiert ist. Für ein symbolisches 3D Objekt ist außerdem das lokale und entfernte visuelle Feedback beim Verschieben des Objektes gezeigt. Die lokale Objektrepräsentation reagiert mit einer Verzögerung, die geringer als die Remote-Verzögerung des Originalobjektes ist. Dadurch ensteht der Eindruck, als ob Server- und Client-Objekt durch ein Gummiband verbunden sind. Durch die Einstellung der Client-Anwendung am Punkt P_{opt} wird dem Gummiband eine optimale Elastizität gegeben, damit der Benutzer sowohl lokal bequem ar-

beiten kann, als auch ein sofortiges Feedback vom Server bekommt. Auf der rechten Seite von Abbildung 5.11 ist die Auswertung der Steigung der LFR-RFR-Kurve aufgetragen, wobei die verschiedenen Kurven zu einer unterschiedlichen Geometriebelastung des Client-Rechners gehören. Für eine anwachsenden Szenengröße am Client verschiebt sich diese Kurve und verändert teilweise ihre Form. Die Kurve wurden durch Interpolation der Meßwerte mit Hilfe des Auswertungsprogramms *Gnuplot* erzeugt.

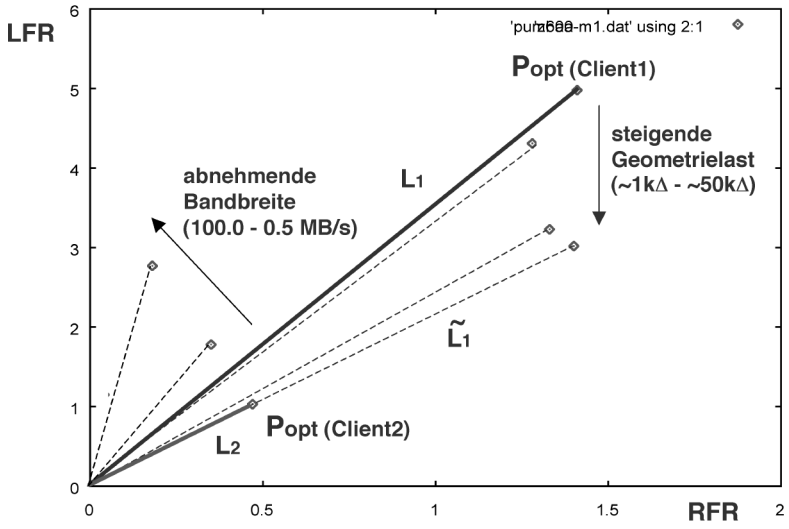

Abbildung 5.12: Ermittlung der Leistung und des Verhaltens eines Client-Rechners bei variierender Geometriebelastung und Verbindungsbandbreite.

In Abbildung 5.12 ist das Verhalten zweier Client-Anwendungen bei der Veränderung der Randbedingungen, d.h. der lokalen Geometrielast und der Anbindungsbandbreite, im LFR-RFR-Diagramm gezeigt. Die Lage eines optimalen Punktes P_{opt} im LFR-RFR-Diagramm wird durch seine Entfernung vom Ursprung $L(LFR, RFR) = \sqrt{LFR^2 + RFR^2}$ und vom Winkel $\alpha(LFR, RFR)$ zwischen der RFR-Achse und der Achse, die durch den Ursprungs und P_{opt} verläuft, angegeben. Die Länge L ist ein Maß für die Leistung des Rechners und der Winkel α erlaubt Angaben darüber zu machen, ob ein Rechner besser für die Remote Visualisierung oder die lokale Visualisierung geeignet ist. Bei der Belastung von Client 1 mit einer größeren Anzahl von Dreiecken verschiebt sich das Verhältnis von LFR und RFR in Richtung der RFR-Achse, wie in Abbildung 5.12 dargestellt. Die lokale Framerate nimmt ab, während die Remote-Framerate im wesentlichen gleich bleibt. Die anfängliche Leistung $L_1 = 5.2$ sinkt auf ein Leistung $\tilde{L}_1 = 3.4$ bei einer Zunahme der Dreiecksanzahl von 1000 auf 50000 Dreiecke in der Client-Szene.

Im Fall von Client 2, der mit $L_2 = 1.1$ eine schwächere Leistung als Client 1 aufweist, wurde die verfügbare Bandbreite von $100MB/s$ schrittweise auf $0.5MB/s$ verringert. Das Verhalten der Client-Anwendung ist in Abbildung 5.12 ebenfalls gezeigt. Die lokale Framerate steigt, während die Remote-Framerate sinkt, da die Client-Anwendung weniger Bilddaten erhält und daher mehr Leistung für die Verarbeitung der lokalen Geometrie aufbringen kann. Dieses Verhalten wird durch die automatische Anpassung der Client-Anwendung nach jeder Performanzauswertung erreicht.

Die Gleichungen zur Bestimmung der Leistung L, des Winkels α und der Größe F, die die

Änderung der Leistung für zwei aufeinanderfolgende Auswertungsschritte widerspiegelt, sind im folgenden angegeben:

$$L(LFR, RFR) = \sqrt{LFR_{opt}^2 + RFR_{opt}^2} \tag{5.3}$$

$$\tan\alpha = \frac{LFR_{opt}}{RFR_{opt}} \tag{5.4}$$

$$F = \Delta LFR * \Delta RFR \tag{5.5}$$

Um das Verhalten zweier Rechner vergleichen zu können, sollte ein Maß für die Änderung der Leistung bei einer festgelegten Zunahme der Belastung gefunden werden, z.B. bei der zusätzlichen Belastung beider Client-Anwendungen durch die gleiche Menge an Dreiecken. Diese Maß wird hier als Güte bezeichnet und folgendermaßen beschrieben. Die Güte G eines Client-Rechners wird durch die drei obigen Werte bestimmt, d.h. $G = G(L, \alpha, F)$. Die Werte ΔLFR und ΔRFR geben dabei die Änderungen der Größen von einem Auswertungsschritt zum nächsten an. Die Gütewerte werden nun folgendermaßen ausgedrückt, wobei der Index ini den Ausgangswert einer Größe vor der erneuten Performanzauswertung meint.

$$G = \frac{L_\alpha}{F} = \frac{1}{(\Delta LFR)(\Delta RFR)}\sqrt{p_\alpha LFR_{ini}^2 + q_\alpha RFR_{ini}^2} \tag{5.6}$$

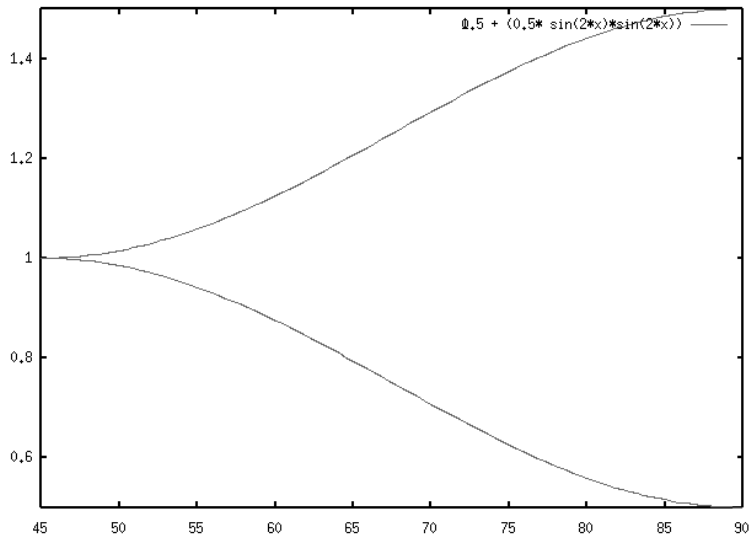

Abbildung 5.13: Verlauf der Gewichtungsfaktoren q_α und p_α.

Um den Einfluß des Winkel α zu berücksichtigen, sind Gewichtungsfaktoren in Gleichung 5.6 für die Güte eingefügt worden, die die jeweiligen Anteile von LFR und RFR am Gesamtwert verstärken bzw. schwächen. Diese Gewichtung soll ausdrücken, daß Client-Anwendungen, die sowohl hohe lokale Frameraten als auch hohe Remote-Frameraten aufweisen, stärker zu bewerten sind

als solche, die lediglich eine hohe lokale Leistung aufbringen. Es wird angenommen, daß die Winkelwerte zwischen 45^o und 90^o variieren können, da für Winkel unter 45^o die Remote-Verzögerung kleiner als die lokale Verzögerung sein müsste. Dieser Fall kann aber praktisch nicht vorkommen, da die Texture-Mapping-Rate nicht größer als die lokale Framerate werden kann. Bei einem Winkel $\alpha = 45^o$ sollen beide Anteile gleich stark gewichtet werden, während mit ansteigendem Winkel gegen $\alpha = 90^o$ der LFR-Anteil schwächer und der RFR-Anteil stärker bewertet werden soll. Eine mögliche Wahl der Gewichtungsfaktoren p_α und q_α ist in den folgenden Gleichungen 5.7 gezeigt. Der Verlauf der beiden Faktoren zwischen $\alpha = 45^o$ und $\alpha = 90^o$ ist in Abbildung 5.13 gezeigt. Bei $\alpha = 45^o$ haben beide Faktoren den Wert 1.0, während für $\alpha = 90^o$ die Faktoren Werte von $q_\alpha = 1.5$ und $p_\alpha = 0.5$ annehmen.

$$q_\alpha = \frac{3}{2} - \frac{1}{2}sin^2(2\alpha) \quad und \quad p_\alpha = \frac{1}{2} + \frac{1}{2}sin^2(2\alpha) \tag{5.7}$$

Die Performanzmetrik PM zum Vergleich von zwei Client-Rechnern wird dann durch die euklidische Distanz der Gütewerte festgelegt:

$$PM = \sqrt{G^2_{Client2} - G^2_{Client1}} \tag{5.8}$$

In den folgenden Kapiteln wird die Einordnung von Client-Rechern in Leistungsklassen mit den hier beschriebenen Größen zur Leistungsbewertung beschrieben. Dazu wird in erster Linie der Leistungswert L verwendet. Die Kontrolle des LoI-Reglers mit Hilfe der Performanzmetrik ist in Kapitel 8 dargestellt.

5.9 Zusammenfassung

In diesem Kapitel wurde das Konzept der Skalierbaren Szenen in einer Gesamtübersicht vorgstellt und mit Ansätzen aus der Literatur verglichen. Darauf wurden die wichtigsten Komponenten des Konzeptes vorgestellt und nacheinander im Detail beschrieben. Der Ansatz der Trennung eines Navigations- und Interaktionsraumes, die durch eine Kombination von lokalem und Remote-Rendering für einen Benutzer zugänglich gemacht werden, und der Level-of-Information Ansatz zum variablen Umgang mit allen Objektrepräsentationen der Visualisierungs-Pipeline in einem verteilten System sind als Kern der Arbeit hervorgehoben wurden. Ein kooperatives Modell zur Mehrbenutzer-Visualisierung, ein Konzept zur Rechteverwaltung im System und eine Performanzmetrik zur Klassifizierung von Client-Rechnern wurden diskutiert und Lösungen für die Software-Architektur SCA3D aufgezeigt. In den folgenden Kapiteln wird nun der Entwurf und die Umsetzung dieser Software-Architektur nach dem Konzept der Skalierbaren Szenen beschrieben. Im weiteren wird dann der Einsatz von SCA3D zur adaptiven 3D Visualisierung von geschützen 3D Dokumenten in Digitalen Bibliotheken und das kooperative Modell untersucht.

Kapitel 6

Entwurf der verteilten Software-Architektur

In diesem Kapitel wird der software-technische Entwurf des Konzeptes aus dem vorhergehenden Kapitel beschrieben. Als Resultate des Kapitels werden ein Analysemodell und ein Designmodell der Software-Architektur SCA3D, die das Arbeiten mit skalierbaren Szenen ermöglicht, angegeben. Anhand der Komponenten und Teilsysteme des Entwurfs wird der Aufbau der Software-Architektur vorgestellt und die Kommunikation innerhalb des Systems erläutert. Mit Hilfe von Diagrammdarstellungen aus dem Umfang der Modellierungssprache UML werden verschiedene Sichten der Software-Architektur beschrieben.

6.1 Einführung: Der Entwicklungsprozeß

Der professionelle Entwurf von Software-Systemen sieht in der Regel vier Phasen vor, nämlich die Analysephase, die Designphase, die Implementierungsphase und zuletzt die Integrationsphase. In der Analysephase werden die Anforderungen an das System in einer Übersicht festgestellt und in Form eines Analysemodells beschrieben. In der Designphase wird die Voraussetzung einer idealen, nicht weiter beschriebene Systemumgebung aufgegeben und ein Designmodell angefertigt, das die Randbedingungen der Rechnerumgebung berücksichtigt. In der Implementierungsphase werden die Bestandteile des Systems angefertigt, z.B. durch Programmierung eines objekt-orientierten Systems in einer gängigen Sprache. In der Integrationsphase wird das System nach möglichst umfangreichen Tests in das Gesamtsystem eingebracht. Dieser Schritt erfolgt nur für große Systeme, die aus vielen getrennt entworfenen Teilsystemen bestehen.

Bei einem iterativen bzw. spiralförmigen Entwurfsprozeß (siehe z.B. [Jac92]) werden nun die vier Phasen von neuem durchlaufen, um so das System immer besser an die tatsächlichen Anforderungen und Randbedingungen anzupassen. In jedem Durchlauf der Spirale entsteht eine funktionsfähiges Teilsystem. In Abbildung 6.1 ist der beschriebene Entwicklungsprozeß dargestellt. Zusätzlich zu den globalen Iterationsschritten sollten nach jeder Implemetierungsphase auch Testschritte durchgeführt werden und die Ergebnisse direkt zur Verbesserung des aktuelle Systems benutzt werden. Bei der Entwicklung der Software-Architektur SCA3D wurde diese Arbeitsweise verfolgt. Der Integrationsphase kam dabei wenig Bedeutung zu, da SCA3D nicht als Teilsystem in ein größeres System eingefügt werden sollte.

Zur Beschreibung der einzelnen Phasen wird die Sprache UML (siehe z.B. [Alh98]) verwendet, die aus einer Zusammenfassung der wichtigsten objekt-orientierten Entwurfsmethoden entstanden

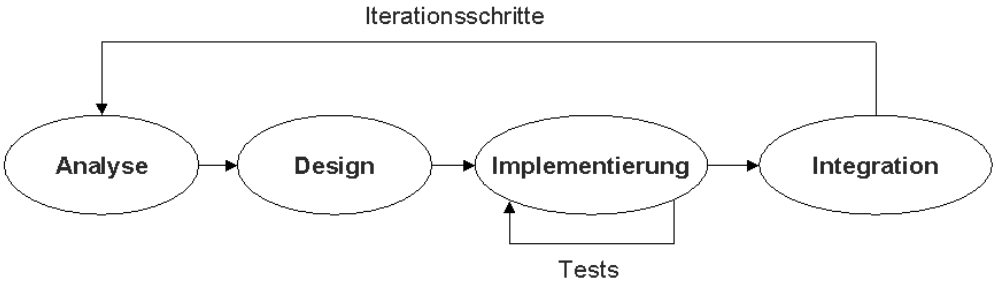

Abbildung 6.1: Iterativer Entwurfsprozeß für Software-Systeme.

ist und heute einen weitverbreiteten Standard darstellt. Die Notation von UML umfaßt Diagramme für die Darstellung der verschiedenen Ansichten auf das Software-System. Ein Modell stellt wichtige Aspekte der Realität dar, beschreibt aber nicht alle Einzelheiten eines realen Systems. Die Verwendung von Diagrammen bieten eine Möglichkeit wichtige Aspekte des Systems herauszuarbeiten und damit dessen Komplexität für den Entwickler zu reduzieren. In diesem Sinn sind die Entwicklungsdiagramme als Ausschnittsansichten eines Gesamtsystems zu sehen. Folgende Diagrammarten werden von UML unterstützt: Use-Case-Diagramme, Klassendiagramme, Interaktiondiagramme, Package-Diagramme, Zustandsdiagramme, Aktivitätsdiagramme und Implementierungsdiagramme. UML bietet zur Erstellung der genannten Diagramme eine festgelegte Menge von Beschreibungselementen. Da nicht alle Diagrammtypen hier verwendet werden, sollen nur die tatsächlich benötigten beschrieben werden.

- *Use-Case-Diagramme*: Use-Case-Diagramme werden in allen vier Phasen der Entwicklung angewendet. Use-Cases wurden zuerst von Jacobson [Jac92] eingeführt und beschreiben das Zusammenwirken von Aktoren, d.h. Personen oder anderen Systemen, mit einem System. Personen werden nach ihrem Rollenverhalten unterschieden, also z.B. Kunde und Administrator. Die Use-Cases selbst sind typische Handlungen, die ein Aktor an einem System ausführt, und werden als Ellipsen in dem Diagramm dargestellt. Durch Linien zwischen den Use-Cases und den Aktoren werden Verbindungen hergestellt, z.B. *uses* oder *extends.*

- *Klassendiagramme*: Klassendiagramme werden vorallem in den Design- und Implementierungsphasen benutzt. Dabei werden die im System vorkommenden Klassen beschrieben, die z.B. durch Verwendung von CRC-Karten (*Class, Responsibility, Collaboration*) ermittelt wurden. UML unterscheidet Klassen und Objekte in den Diagrammen nicht. Im Klassendiagramm werden die Klassen durch ein Symbol vertreten, das den Namen, die enthaltenen Attribute und die Methoden enthält. Die Symbole der einzelnen Klassen werden durch Verbindungen in Beziehung zueinander gesetzt. Die wichtigsten Beziehungen sind Assoziation (allgemeine Beziehung), Aggregation (Ist-Teil-von-Beziehung), Komposition (Enthaltensein-in-Beziehung) und Vererbung (Ist-ein-Beziehung).

- *Interaktionsdiagramme*: Diese Diagramme, die den zeitlichen Programmablauf als Aufrufsequenzen angeben, werden in allen vier Phasen der Entwicklung eingesetzt. Es werden weiter Sequenzdiagramme und Kollaborationsdiagramme unterschieden. Bei den Sequenzdiagrammen werden die Klassensymbole horizontal aufgereiht und von jeder Klasse eine senkrechte

Linie, die die Zeitachse darstellt, nach untern gezeichnet. Zwischen diesen Linien werden Aufrufe zwischen Klassen durch Pfeile dargestellt. Das Kollaborationsdiagramm unterscheidet sich nur durch die Darstellung, die keine senkrechten Zeitachsen verwendet, sondern den Ablauf durch Sequenzen von Klassen darstellt [Wah98].

6.2 Anforderungsanalyse: Komponenten und ihre Modellierung

In diesem Abschnitt soll eine Analyse der Anforderungen an die Software-Architektur SCA3D erstellt werden. Dazu wird zuerst ein Use-Case-Diagramm angefertigt, um einen Überblick über die Funktionalität des Systems zu erhalten. Dann werden die Klassen bzw. Objekte im Problemraum des Konzeptes identifiziert und in Beziehung zueinandergesetzt. Zusammen mit dem Use-Case-Diagramm wird daraus das Analysemodell erzeugt. Hinzukommen können noch Sichten auf das System, z.B. in Form von prototypischen Benutzerschnittstellen. In der folgenden Auflistung sind noch einmal die wichtigsten Anforderungen an ein System genannt, das dem Konzept der Skalierbaren Szenen entspricht. Die Anforderungsanalyse ist in drei Hauptbereiche eingeteilt, nämlich digitale 3D Dokumente, offenen Informationsräume und Digital-Library-Systeme (DL-System), um eine Vorstrukturierung der Anforderungen an das Systems zu erlauben. Die SCA3D-Architektur wird als ein solches DL-System entworfen. Die allgemeine Anforderungen an die Software-Architektur SCA3D sehen wie folgt aus.

Anforderungen an digitale 3D Dokumente:

- Aufbau: Struktur, Inhalt und Metadaten
- Dokumentschnittstellen für: visuelles Cut-and-Paste, Indizierung, Erweiterbarkeit, Sicherheitsmanagement
- Anpassbarkeit der Komplexität: Level-of-Information-Ansatz

Anforderungen an offene Informationsräume:

- Einheitliche Bedingungen für das Arbeiten mit digitalen Objekten: Navigations- und Interaktionsräume
- Schnittstellen zu anderen offenen Informationsräumen
- Möglichkeiten zum Sicherheitsmanagement

Anforderungen an DL-Systeme:

- Erfüllung der Charakteristiken von offenen Informationsräumen
- Unterstützung von digitalen 3D Dokumenten: objekt-orientiertes, internes Dokumentmodell, Kommunikation zur Unterstützung des Level-of-Information-Ansatzes
- Möglichkeiten zur Erkundung bestehender DL Dokumente (Retrieval, Darstellung, Interaktion), zum Generieren und Hinzufügen neuer Dokumente
- Bewertungsmöglichkeit für die Güte (Leistung und Anbindung) von Benutzerrechnern
- Kooperationsmöglichkeiten: Benutzerverwaltung, Synchronisation, Awareness

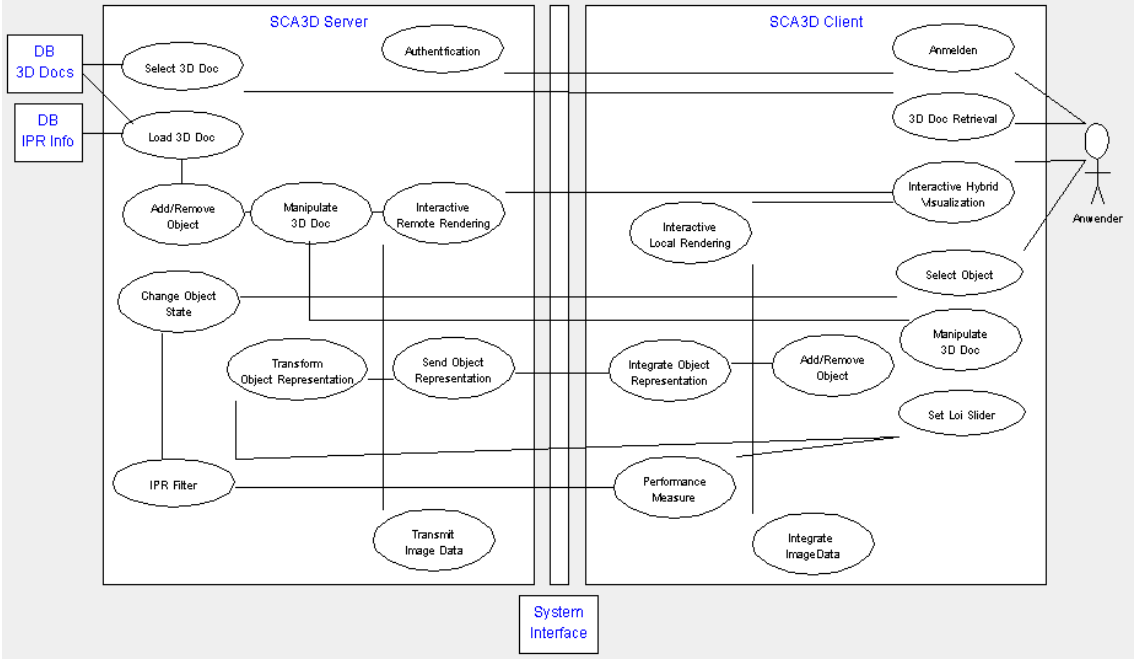

Abbildung 6.2: Erweitertes Use-Case-Diagramm für SCA3D.

Für das SCA3D-System, das entsprechend diesen Anforderungen entwickelt wurde, ist in Abbildung 6.2 ein erweitertes Use-Case-Diagramm dargestellt. Der Anwender ist der Hauptakteur des Systems. Er bedient eine Client-Applikation, die über eine Systemschnittstelle mittels eines Netzwerkes mit dem Server verbunden ist. Der Server wiederum ist mit Datenbanken verbunden, die die digitalen Dokumente und die dazugehörigen Sicherheitsinformationen enthalten. Die wichtigsten Aktionen sind in diesem Szenario durch ovale Symbole repräsentiert. Die Aktionen, die zwischen Client und Server ablaufen, werden mit Hilfe eines Middleware-Interfaces übertragen. Außerdem werden Daten über Socket-Verbindungen (Bilddaten-Interface) und HTTP-Verbindungen (WWW-Interface) übertragen.

Dem Benutzer wird lokal ein 3DVis-GUI (*Graphical User Interface*) für die interaktive 3D Visualisierung zur Verfügung gestellt und eine Panel-GUI, das die Kontrollelemente, wie z.B. LoI-Schieberegler und Performance-Anzeige, enthält. Weiterhin können die Server-Komponenten der Software-Architektur über eine WWW-Schnittstelle kontrolliert werden, um z.B. den Server zu starten bzw. zu stoppen oder auf die Datenbank zuzugreifen. Die wichtigsten Handlungen eines Benutzers sind in Abbildung 6.2 direkt mit dem Akteur verbunden: Anmelden, 3D Dokument-Retrieval, interaktive 3D Visualisierung, Objektselektion, Objektmanipulation und Positionierung des LoI-Schiebereglers. Die Handlungen im Zusammenhang mit der externen Datenbank sind: 3D Dokument-Auswahl und 3D Dokument-Laden. Die weiteren Handlungen in Abbildung 6.2 laufen intern und zwischen Client und Server ab.

Um in der Analysephase einen Eindruck von den Benutzerschnittstellen des Systems zu bekommen, werden diese als Sichten graphisch dargestellt. Es handelt sich bei diesen Sichten um GUI-Prototypen ohne dahinterliegende Funktionalität. In Abbildung 6.3 sind die Benutzerschnittstellen 3DVis-GUI und Panel-GUI gezeigt. Das 3DVis-GUI dient als Schnittstelle zur interaktiven

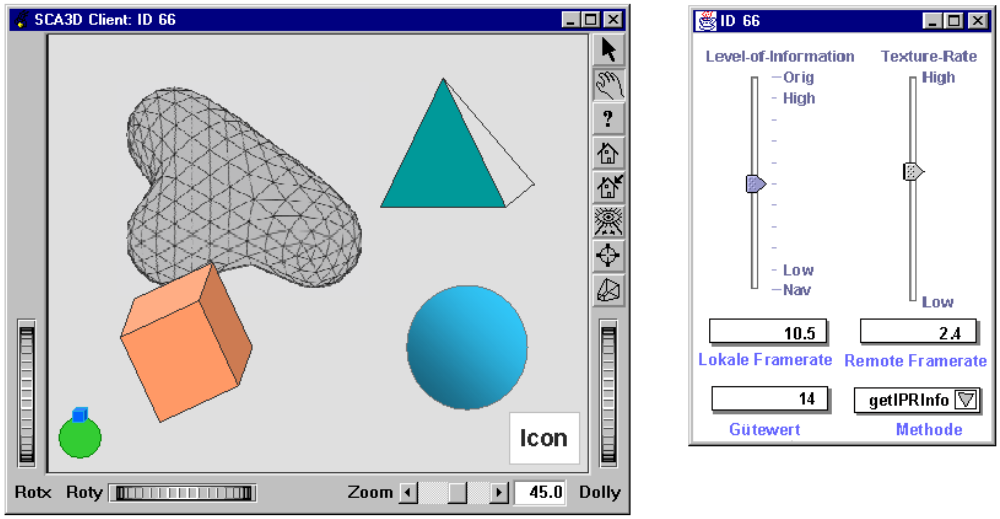

Abbildung 6.3: Sicht der Benutzerschnittstellen 3DVis-GUI (links) und Panel-GUI (rechts).

Visualisierung der 3D Dokumente, während das Panel-GUI die verschiedenen Schieberegler der Client-Applikation, Anzeigefelder für lokale Framerate, Remote-Frameraten bzw. Gütewerte und ein Auswahlfeld für Methoden (z.B. getIPRInfo oder Retrieval) bereitstellt. Kontrollelemente im Visualisierungsfenster des 3DVis-GUI geben Informationen über den aktuellen Navigationsmodus für den Mehrbenutzerbetrieb an (farbiges 3D Symbol im linken unteren Bereich) und über die Teilnehmer, die durch Logos im unteren rechten Bereich dargestellt werden. Neben dem Level-of-Information-Regler ist hier auch ein Regler zur Kontrolle der lokalen Texture-Mapping-Rate eingefügt, der zur Einstellung des Verhältnisses von lokaler und Remote-Verzögerung dient.

6.3 Designmodell: Verteilter Entwurf für das Konzept der skalierbaren Szenen

Nachdem die Anforderungsanalyse abgeschlossen ist, wird nun die Systemumgebung in den Entwurf miteinbezogen. Zu diesem Zweck wird ein Designmodell erstellt, das die oben ermittelten Klassen und Subsysteme näher beschreibt und auch die Verteiltheit des Systems widerspiegelt. Die Designphase beginnt daher, sobald die Randbedingungen der Implementierung berücksichtigt werden. Die Software-Architektur SCA3D ist eine Client-Server-Architektur und wird als objekt-orientiertes System umgesetzt. Der allgemeine Aufbau enthält ein Server-System, ein Client-System und eine Middleware-Schnittstelle zwischen diesen beiden Hauptkomponenten. Da das System in der späteren Implementierungsphase mit der objekt-orientierten Sprache Java, der Middleware CORBA und weitgehend plattformunabhängig realisiert werden soll, werden im folgenden diese Randbedingungen beim Design berücksichtigt.

Die wichtigsten Bestandteile des Gesamtsystems werden nun mit Hilfe verschiedener Diagramme näher erläutert. Zur Strukturierung der verschiedenen Sichtweisen auf die Architektur, d.h. der verwendeten Diagramme, ist in Abbildung 6.4 das Package-Diagramm der SCA3D Architektur dargestellt. Package-Diagramme geben die Hauptkomponenten eines Systems und deren

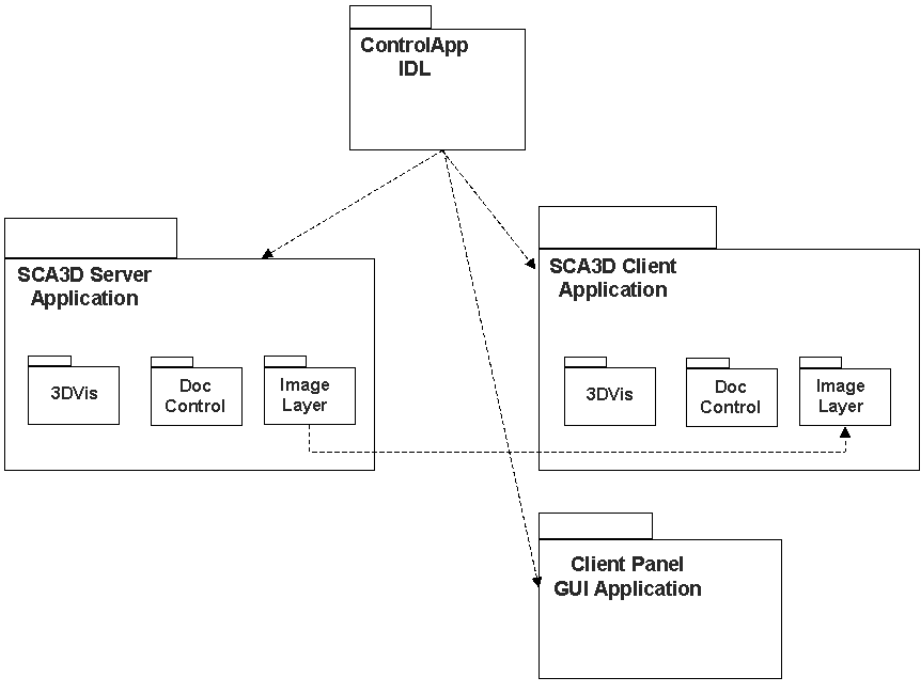

Abbildung 6.4: Package-Diagramm der SCA3D Architektur.

Abhängigkeiten bei Änderung einer Komponente an. Hier sind die wichtigsten Komponenten die SCA3D Server-Applikation, die SCA3D Client-Applikation, das ControllApp-IDL-Interface und das Client-Panel. Die Abhängigkeiten, die durch unterbrochene Linien gekennzeichnet sind, bestehen zwischen der ControllApp-Komponente und allen anderen Komponenten. Außerdem besteht eine direkte Abhängigkeit zwischen Server und Client durch die Bildkommunikationsschicht, so daß bei Änderungen an der Bildübertragung in den meisten Fällen Client- und Server-Klassen angepaßt werden müssen. Innerhalb des Server- und Client-Package sind jeweils weitere Packages für 3D Visualisierung, Dokumentkontrolle und Bilddatenkommunikation eingetragen, die die Hauptkomponenten der jeweiligen Anwendung ausmachen.

6.3.1 Die Visualisierungskomponenten

Die Komponenten zur 3D Visualisierung bestehen aus einer server-seitigen und einer client-seitigen Visualisierungsanwendung, die jeweils einen interaktiven Szenengraph verwalten. Zum Aufbau dieser Visualisierungskomponenten wird das OpenInventor-API verwendet. Nachdem ein Basisszenengraph aufgebaut ist, der durch Einlesen externer 3D Dokumente in entsprechende Szenengraphobjekte entsteht, wird dieser an die Instanz einer Offscreen-Renderer-Klasse bzw. einer Viewer-Klasse übergeben. Im ersten Fall werden Bilddaten in einen internen Buffer abgelegt, die weiterverarbeitet werden können, im zweiten Fall werden Bilder direkt auf dem Benutzerbildschirm ausgegeben. Weiterhin werden Aktionen des Benutzers ausgewertet und als Eingabe auf den Szenengraph angewendet. In Abbildung 6.5 sind diese Komponenten dargestellt.

Beide Visualisierungsanwendungen werden verbunden, indem die Bilddaten, die am Server vom Offsreen-Renderer in den Bildbuffer geschrieben werden, am Client durch Texture-Mapping

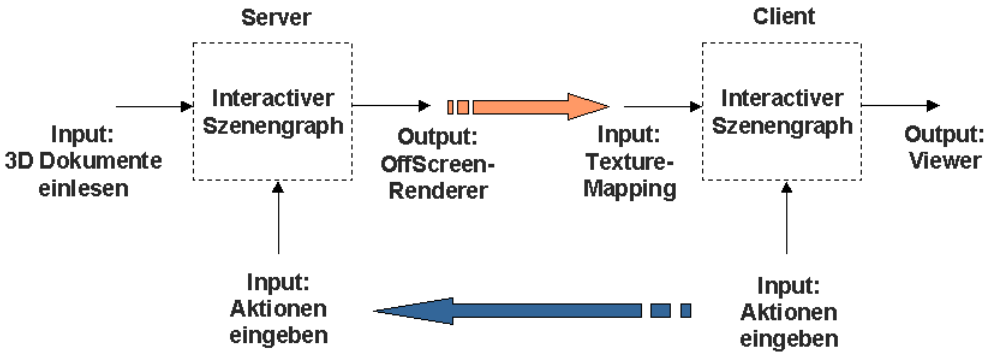

Abbildung 6.5: Komponenten der 3D Visualisierungsschicht.

in den lokalen Szenegraph integriert werden. Die Synchronisation der Benutzeraktionen von Client-Anwendung und Server-Anwendung schließt den interaktiven Visualisierungskreis. Durch Austausch von 3D Objekten zwischen den Szenengraphen wird das interaktive Arbeiten mit dem Server-Dokument ermöglicht. Der Entwurf des dazu verwendeten Document-Request-Broker-Ansatzes wird im folgenden Abschnitt beschrieben.

6.3.2 Modellierung eines Document-Request-Brokers

Der in Abschnitt 5.5 vorgestellte Ansatz eines Document-Request-Brokers für verteilte 3D Dokumente soll nun im Detail beschrieben werden. Dieser Ansatz nutzt den Aufbau eines 3D Dokumentes, der eine interne Abbildung auf ein objekt-orientiertes Dokumentmodell (OpenInventor-Szenengraph) erlaubt, um mit Teilgruppen des Dokumentes zu arbeiten.

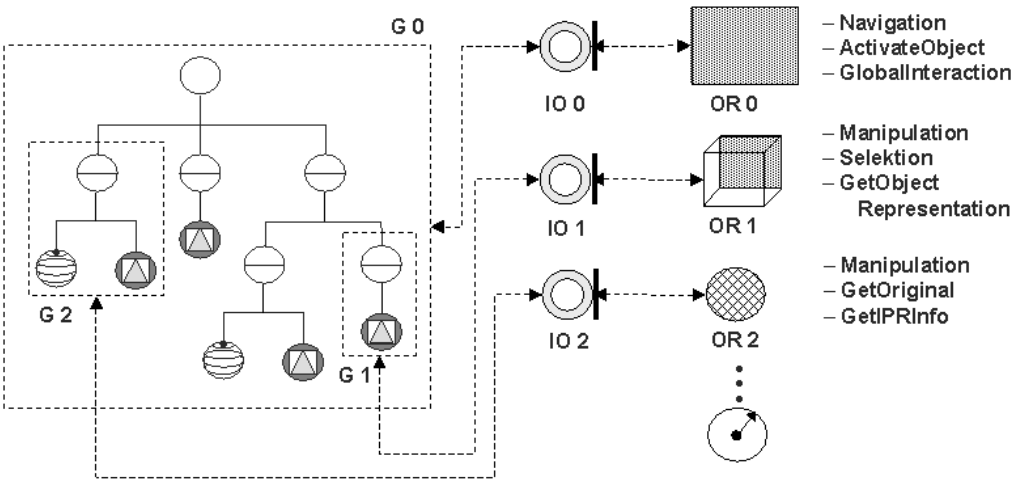

Abbildung 6.6: Schematischer Aufbau des Document-Request-Brokers.

In Abbildung 6.6 ist der Aufbau des Document-Request-Brokers schematisch gezeigt. Das 3D Dokument selbst stellt dort die übergeordnete Gruppe $G0$ dar, die am Beginn einer Visualisierungssitzung dem Benutzer als visuelles Interface durch Remote-Rendering der Gesamtszene zur Verfügung gestellt wird. Durch Selektion einer Bildschirmposition kann nun eine Untergruppe des Dokumentes $G1$ in den Zustand eines aktiven Objektes gebracht werden. Dazu wird auf Server-Seite eine Objekt-Repräsentation $OR1$ und ein Interface-Objekt $IO1$, das als Verbindung zwischen dem Originalobjekt im Server-Szenengraph und der Objekt-Repräsentation dient, erzeugt. Die Klasse der Interface-Objekte in der SCA3D-Architektur ist von der Klasse $SoSeparator$ des verwendeten Visualisierungs-API abgeleitet, sodaß es alle Methoden eines Gruppenobjektes erbt. Die erzeugte Objekt-Repräsentation wird dann zum Client übertragen und in den lokalen Szenengraph eingefügt. Durch Verwendung eines Namensschemas, das die Identifizierung des Interface-Objektes für ein aktives Objekt und dessen Benutzer zuläßt, können nun client-seitige Methodenaufrufe an das Interface-Objekt weitergeleitet und auf das Originalobjekt angewendet werden. Diese Methoden können unter anderem die folgenden Operationen auslösen, wie in Abbildung 6.6 dargestellt:

- *Für das Gesamtdokument* $G0$: ChangeCameraParameters (Navigation), ActivateObject, ChangeGlobalParameters.
- *Für Teilgruppen* $G1$-Gn: InteractObject (Manipulation), SelectObject, GetObjectRepresentation, GetIPRInfo usw.

Um den Ablauf beim Aktivieren eines Objektes und beim Absetzen eines Remote-Document-Calls (RDC) zu erläutern, ist in Abbildung 6.7 ein Sequenzdiagramm dieser Vorgänge gezeigt. Die Klassen, die daran beteiligt sind, sind im Sequenzdiagramm eingezeichnet und ebenso die Methodenaufrufe zwischen ihnen. Der zeitliche Ablauf beim Aktivieren eines Objektes, das vorher im Zustand eines inaktiven Server-Objekts ist und nun in den Zustand eines aktiven Objektes übergeht, wird durch das Sequenzdiagramm erläutert. Durch Auswahl einer Bildschirmposition im Viewer-Fenster mit der Methode $select()$ selektiert der Benutzer eine Teilgruppe des Server-Dokumentes, für die am Server ein Interface-Objekt und am Client eine Objektrepräsentation angelegt wird.

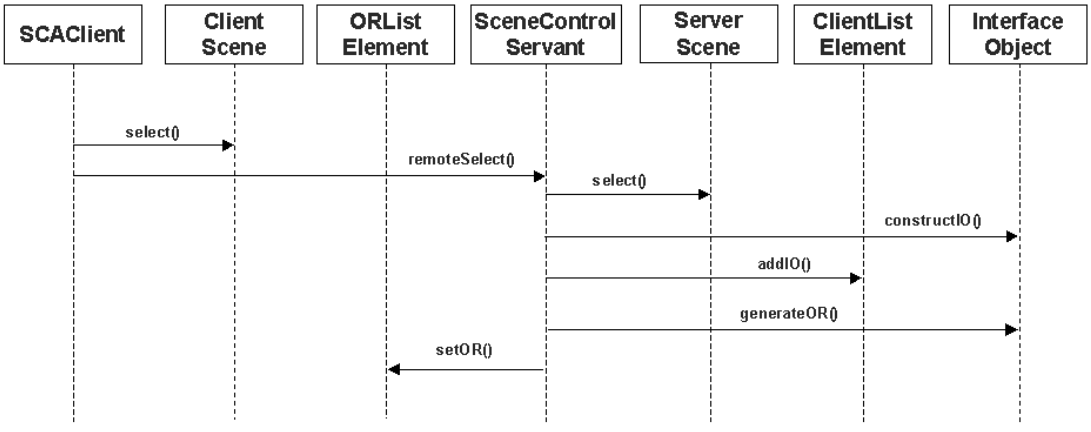

Abbildung 6.7: Sequenzdiagramm für die Aktivierung eines Objektes und die Erzeugung eines Interface-Objektes.

Zu diesem Zweck wird auf Client-Seite nur die Remote-Rendering-Information der Server-Szene verwendet. Auf Server-Seite wird die Selektionsinformation, d.h. die Bildschirmposition mit den Kameraparametern, dazu verwendet, die Selektionsmethode $select()$ auf das Originaldokument anzuwenden. Für das ausgewählte Objekt wird die Pfadinformation extrahiert und für dieses Objekt ein Interface-Objekt mit der Methode $constructIO()$ erzeugt. Dieses Interface-Objekt wird dem Benutzer zugeordnet, indem es in die zugehörige Instanz der Klasse $ClientListElement$ mit der Methode $addIO()$ eingetragen wird. In einer Liste der Instanzen von $ClientListElement$ werden die Benutzerinformationen verwaltet. Mittels der Methode $generateOR()$ des Interface-Objektes wird nun eine passende Objektrepräsentation OR erzeugt und mit einem eindeutigen Namen verknüpft. Diese Objektrepräsentation wird an den Client gesendet und mit der Methode $setOR()$ in die Liste der Objektrepräsentationen und in die Client-Szene integriert. Das Objekt befindet sich nun im Zustand eines aktiven Objektes und der Benutzer kann über seine Objektrepräsentation auf das server-seitige Objekt zugreifen. Zum Aufruf einer Methode wird der Methodenname und die Parameter an das entfernte Objekt übergeben und dort ausgeführt. Das dazu verwendete Namensschema und die entfernten Methodeaufrufe, hier als *Remote-Document-Calls* bezeichnet, werden in den folgenden beiden Abschnitten erläutert.

Das DRB-Namensschema

Um Objektreferenzen im System bekannt zu machen, wird eine Namenschema verwendet, das den Status des jeweiligen 3D Objektes und die Kennnungen des Interface-Objektes und der Client-Anwendung enthält. Inaktive Objekte auf Server-Seite werden durch eine Namen gekennzeichnet, der mit $S0$ beginnt. Inaktive Objekt auf Client-Seite erhalten Namen, die mit $S2$ anfangen. Jedes 3D Objekt im aktiven Zustand erhält einen eindeutigen Namen, der nach dem Schema $S1-ID1-...-IDn$ aufgebaut ist. Dabei bedeutet der erste Teil $S1$, daß sich das Objekt im Zustand eines aktiven Objektes befindet. Die Namensteile IDn beziehen sich auf die Client-Anwendung, die dieses Objekt aktuell interaktiv bearbeitet, also z.B. $ID116$ für die Client-Anwendung mit der Nummer 116. Die zugehörigen Interface-Objekte werden nach dem Schema $IOm - IDn$ benannt und enthalten eine Referenz auf das originale 3D Dokument. Der erste Namensteil IOm gibt dabei die Nummer des Interface-Objektes in der Liste aller Interface-Objekte dieses Benutzers an.

Die eindeutigen Namen werden beim Erzeugen der Interface-Objekte als Metadaten mit dem Objekt verknüpft und bei der Übertragung zum Benutzer den Objektrepräsentationen zugeordnet. Für jede Client-Anwendung wird am Server eine eigene Liste mit Referenzen auf die Interface-Objekt verwaltet. Beim Aufruf einer Methode wird über eine client-seitige Liste der Objektrepräsentation die passende Objektreferenz am Server ermittelt und die Methode für des gefundenen Objektes ausgeführt.

Remote-Document-Calls und SCA3D-Nachrichten

Das Prinzip der hier vorgestellten Remote-Document-Calls ist der Aufruf von Methoden eines entfernten Objektes im Server-Szenengraph über Objektrepräsentationen in der Client-Szene. Die ausführbaren Methoden sind einerseits die von dem Visualisierungs-API bereitgestellten und andererseits zusätzliche Methoden die vom angelegten Interface-Objekt angeboten werden. Beispiele für die zweite Kategorie sind z.B. die Methoden $getObjectRepresentation()$ (Level-of-Information) oder $getIPRInfo()$ (Sicherheitsmanagement).

Zum entfernten Aufruf der entsprechenden Methode wird eine allgemeine Invoke-Methode des Document-Request-Brokers verwendet, die am Server von der Klasse $SceneControlServant$ im-

plementiert wird. Zur Übermittlung des Remote-Document-Calls wird der Invoke-Methode eine Instanz der Klasse $SCAMessage$ als Eingabeparameter übergeben. In der Klasse $SCAMessage$ sind alle wichtigen Informationen für den entfernten Methodeaufruf gekapselt.

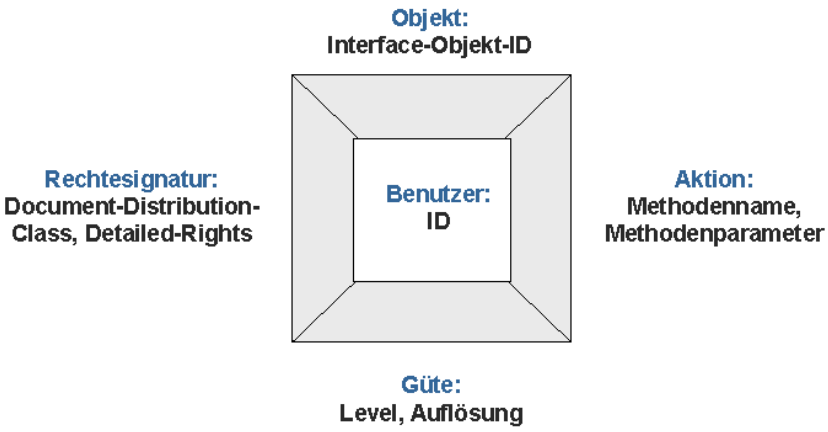

Abbildung 6.8: Aufbau einer SCA3D-Nachricht.

In Abbildung 6.8 ist der Aufbau einer SCA3D-Nachricht schematisch gezeigt. Die Nachricht enthält vier Hauptkomponenten, die Auskunft über das aufgerufene Objekt, die geforderte Aktion, die Güte der Client-Anwendung und die aktuelle Rechtesituation zwischen Benutzer und Server geben. Jede Komponente kann weitere Daten enthalten, die dort jeweils angegeben sind. Die Eingabeparameter der Methode werden mit Hilfe einer allgemeinen Parameter-Klasse angegeben und am Server ausgewertet, sodaß beliebige Kombinationen von Eingabeparametern möglich sind. Die Rückgabeparameter der Operation werden wiederum als Instanz der Klasse $Parameter$ zur Client-Anwendung zurückübertragen.

Im Abbildung 6.9 ist der Ablauf beim Aufruf eines Remote-Document-Calls (RDC) dargestellt. Ein Remote-Document-Call kann an jedes aktive 3D Objekt gesendet werden und umfaßt die oben angegebenen Methoden. Welche Methode beim Absetzen eines RDC ausgeführt wird, wird durch die lokale Anwendung mittels der Methode $determineMethod()$ ermittelt, indem die Eingaben des Benutzers und die Position des Level-of-Information-Reglers ausgewertet werden. Bei einem Selektionsereignis wird überprüft, ob die Objektrepräsentation eines aktiven Objektes ausgewählt wurde. Wenn dies zutrifft, wird die bestimmte Methode mit der Kennung des Interface-Objektes und den Parametern durch Aufruf der Methode $invokeRDC()$ der Objektrepräsentation und der Methode $invoke()$ des $SceneControlServant$-Objektes am Server aufgerufen. Die Kennung des Interface-Objektes ist in der übergebenen SCA3D-Nachricht enthalten. Am Server wird nun mit der Methode $findIO()$ das passende Interface-Objekt gefunden und die beabsichtigte Methode durch Aufruf von $applyMethod()$ auf das entsprechende Interface-Objekt des 3D Dokumentes angewendet. Die Ergebnisse der Operation werden zum Client zurückübertragen.

Um die gewünschte Methode des Interface-Objektes aufrufen zu können, werden im Falle eines RDC alle Methodennamen des betreffenden Interface-Objektes mit der String-Version der vom Client aufgerufenen Methode verglichen. Das Interface-Objekt selbst enthält Referenzen auf die aktive Teilgruppe des Originaldokumentes und zur Ausführung der RDC-Methoden wird auf diese zugegriffen.

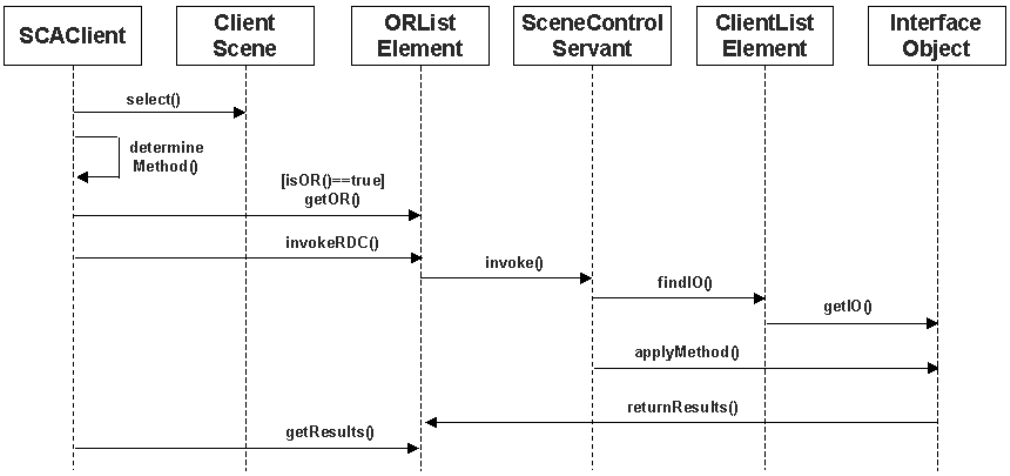

Abbildung 6.9: Sequenzdiagramm für den Aufruf eines Remote-Document-Calls.

6.3.3 Die Level-of-Information-Komponenten

In diesem Abschnitt werden die Komponenten, die für die Modellierung des Level-of-Information-Ansatzes verwendet werden, beschrieben. Diese Komponenten bauen auf dem Document-Request-Broker-Modell auf und stellen Funktionalität zur Transformation und Distribution von Objektrepräsentationen in verteilten Umgebungen zur Verfügung. Hinzu kommen Klassen zur Performanzmessung im System, sodaß jedem Benutzer Objektrepräsentationen zur Verfügung gestellt werden können, die seinen Bedingungen am besten entsprechen.

Transformation von Objektrepräsentationen

Bei der Aktivierung eines Dokumentobjektes wird automatisch eine Objektrepräsentation in der Client-Szene angelegt. Die Repräsentationsform des Objektes wird durch die Position des Level-of-Information-Reglers der Client-Anwendung bestimmt. Dies geschieht durch Aufruf der Methode $getOR()$ mittels eines RDC mit einer SCA3D-Nachricht als Eingabeparameter, die die aktuelle Reglerposition enthält. Auf Server-Seite wird der aktuelle LoI ausgewertet, um die geeignete Objektrepräsentation zu erzeugen. Diese wird dann zur Client-Anwendung übertragen und mittels einer Push-Operation vom Server in den Client-Szenengraph eingefügt. Der Aufbau der Methodenschnittstelle eines Interface-Objektes und die Generierung von Objektrepräsentationen sind im folgenden näher beschrieben:

- *Schnittstelle des Interface-Objektes*: Die Methoden der Schnittstelle des Interface-Objektes können in drei Gruppen eingeteilt werden. Dies sind die Gruppe der Methoden zur Initialisierung des Interface-Objektes, die Gruppe zur Abfrage von Informationen über das Dokumentobjekt und schließlich eine Gruppe zur Transformation des Dokumentobjektes in neue Objektrepräsentationen. Da die Klasse der Interface-Objekte von der OpenInventor-Klasse $SoSeparator$ abgeleitet ist, erbt eine Instanz auch alle Methoden dieser übergeordneten Klasse. In der Initialisierungsgruppe der Methoden eines Interface-Objektes ist ne-

ben den ererbten Methoden, wie z.B. $setName()$ zur Namensgebung, noch die Methode $copyChildren()$ enthalten, die es erlaubt, eine Kopie aller Unterknoten eines Dokumentobjektes als Kopie an das Interface-Objekt anzuhängen. Dadurch wird es möglich veränderte Objektrepräsentationen des Originalobjektes zu erzeugen und zu verteilen, ohne das Originaldokument zu verändern.

Die zweite Gruppe von Methoden umfaßt alle Methoden, die benötigt werden, damit eine Benutzer Informationen über das Originaldokument am Server abfragen kann. Diese Informationen, wie z.B. Dokumentkennung, Komponentennamen, IPR-Daten, sind als Metadaten an das Dokument angehängt. Die zugehörigen Methoden sind z.B. $getDocID()$, $getModelNames()$ oder $getIPRInfo()$. Beim Aufruf durch einen Benutzer über einen RDC werden diese Metadaten am Server ausgelesen und von der Client-Anwendung ausgegeben, z.B. in Form einer Textausgabe in ein eigenes Fenster.

- *Generierung von Objektrepräsentationen*: Die dritte Methodengruppe besteht aus den Methoden zur Transformation von Objekten. Die wichtigste Methode ist $getOR(Level-of-Information)$, mit der die Objektrepräsentation erzeugt wird, die den angegebenen Level-of-Information-Daten entspricht. Diese LoI-Daten beschreiben den Level und die Auflösung der Objektrepräsentation, die von einer Client-Anwendung angefordert wird. Innerhalb der Methode wird nun diese Information ausgewertet und ein entsprechendes 3D Objekt erzeugt. Wenn der Original-Level eingestellt ist, werden die Originaldaten direkt ausgegeben.

  Bei einem Level, der 3D-Netzen entspricht, wird aus dem Originalobjekt ein 3D-Netz mit der angegebenen Auflösung erzeugt. Der niedrigste Level, der nur Navigation im server-seitigen Dokument erlaubt, bewirkt, das das selektierte Objekt in der Server-Szene durch eine eingefärbte Bounding-Box markiert wird, sodaß diese Markierung am Client als Bildinformation sichtbar wird. Nachdem eine Objektrepräsentation erzeugt wurde, wird diese im Austauschformat (OpenInventor-Format) als String ausgegeben und mittels einer remote-zugänglichen Methode der Client-Anwendung, die vom Server aufgerufen wird, in die Client-Szene integriert.

  Zur Generierung der Objektrepräsentationen werden OpenInventor-Methoden verwendet. Als Beispiel sei die 3D-Netzerzeugung beschrieben. Zur Erzeugung von 3D-Netzen wird mit Hilfe einer Aktion ($SoCallBackAction$) ein Callback ($SoTriangleCB$) auf den betreffenden Objektknoten angewendet, wobei durch Eckpunktbestimmung ein Dreiecksnetz mit einer festzulegenden Auflösung erzeugt wird. Dieses Netz wird dann an einen $SoSeparator$-Knoten angehängt und als Rückgabe der Methode $getOR()$ ausgegeben.

Performanzmessung und -auswertung

Die Performanzmessung beruht auf der Bestimmung von lokaler und Remote-Framerate. Außerdem wird die Größe der Client-Szene benötigt, um die Belastung der Client-Anwendung einordnen zu können. Zu diesem Zweck ist auf Client-Seite eine $Performance$-Klasse angelegt, die zur Bestimmung von lokaler Framerate und Szenengröße Methoden bereitstellt. Die Steuerung der Messung und die Anzeige der Meßergebnisse geschieht mit Hilfe des Client-Panel-GUI, das in Abbildung 6.3 als Sicht auf die Benutzerschnittstelle von SCA3D gezeigt ist.

Die lokale Framerate wird ermittelt, indem bei jeder Rendering-Aktion ein Zähler heraufgesetzt wird, der die Bildanzahl repräsentiert. Durch gleichzeitige Messung einer Start- und Stopzeit wird für einen Mittlungszeitraum die lokale Framerate aus dem Verhältnis von Bildern pro Zeiteinheit

bestimmt. Um die maximal mögliche Framerate bestimmen zu können, muß sich die Szene oder die Kamera bewegen, anderenfalls wäre die so gemessene Framerate gleich Null. Daher wird für eine Messung der Leistung des Systems eine künstliche Bewegung der Kamera vorgenommen. Wenn ein Dokument dynamische Komponenten enthält oder der Benutzer während der Messung mit dem Dokument ständig interagiert, ist dies nicht nötig. Die Größe der Client-Szene wird durch Anwendung einer $CountTriangleCB$-Aktion des Openinventor-API gemessen. In Abbildung 6.10 sind die Abläufe während einer Performanzmessung einer Client-Anwendung als Sequenzdiagramm dargestellt. Die Messung wird nicht kontinuierlich durchgeführt, sondern beim Anmelden und dann in bestimmten Zeitabständen aufgerufen. Die Ergebnisse werden im Client-Panel-GUI angezeigt.

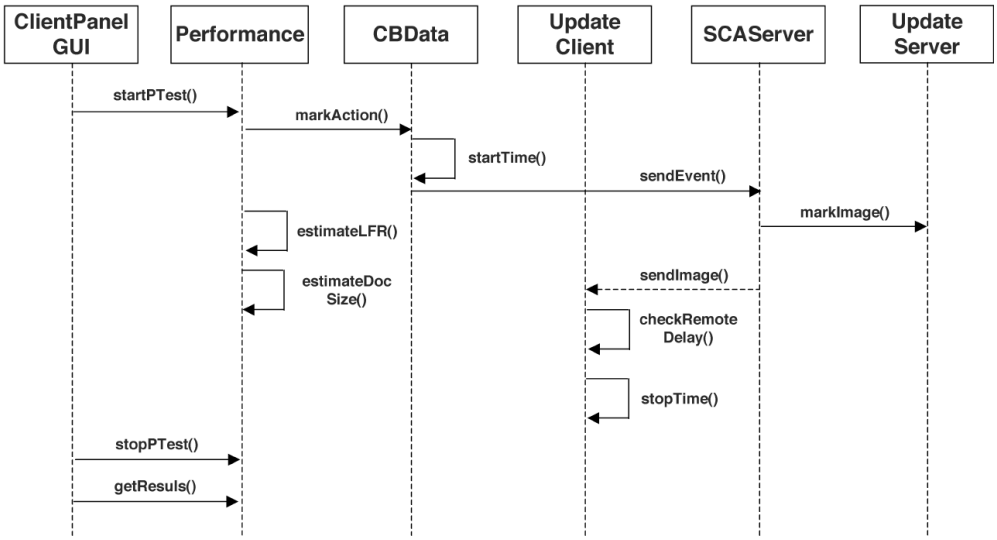

Abbildung 6.10: Sequenzdiagramm der Abläufe beim Performanztest einer Client-Anwendung.

Die Bestimmung der Remote-Framerate geschieht, wie in Abschnitt 5.8.1 angedeutet, durch zeitliche Markierung von Aktionen und den dazu gehörigen Bilddaten. Dazu wird am Client einer Aktion eine eindeutige Kennung zugeordnet und zusammen mit dem Startzeitwert gespeichert. Solange diese Einzelmessung läuft wird keine neue Aktion markiert. Die Aktion wird als Event zum Server übertragen und dort auf das Dokument angewendet. Die mitgesendete Aktionskennung und die Client-Kennung wird nach der Ausführung in das nun gerenderte Bild als markierter Pixel eingebracht. Dies geschieht durch die Methode $TextureCB()$ der Update-Klasse am Server, die für die Erzeugung der Bilddaten durch Offscreen-Rendering zuständig ist. Auf Client-Seite wird, während eine Einzelmessung läuft, jedes Bild auf den Zustand des Markierungspixels hin untersucht. Hierfür wird die Methode $checkRemoteDelay()$ der Client-Klasse $Update$ verwendet, die das Bild als Textur auf die Projektionsfläche der Client-Szene abbildet. Wenn die aktuelle Aktionskennung und die eigene Client-Kennung im Bild vorhanden ist, wird die Stopzeit der Aktion bestimmt und daraus die Gesamtdauer der Aktionsausführung und damit die Remote-Verzögerung errechnet.

Da die Texture-Mapping-Rate, mit der Bilddaten auf die Projektionsfläche der Client-Szene abgebildet werden, variable regelbar sein soll, kann sie durch die Methode $setTMSlider()$ der Klasse $CPanelServant$ dynamisch eingestellt werden. Die Regulierung erfolgt entweder über das Client-Panel-GUI durch den Benutzer oder automatisch durch die Client-Anwendung. Um die

server-seitige Rendering-Rate an die Texture-Mapping-Rate einer Client-Anwendung anzupassen, kann über eine remote-aufrufbare Methode des Servers die Server-Framerate vom Client aus eingestellt. Durch eine Regelkreis paßt sich das System so an, daß die Texture-Mapping-Rate möglichst der optimalen Remote-Framerate entspricht.

Modellierung des IPR-Filters zum Sicherheitsmanagement

Es wird davon ausgegangen, daß die benötigten Metadaten, hier kurz mit IPR-Information (*Intellectual Property Rights*-Information) benannt, zum Sicherheitsmanagement in einer Datenbank am Server vorliegen. Beim Einlesen eines Dokumentes durch den Server eines digitalen Bibliothekssystems werden diese Informationen in das interne Dokumentmodell integriert. An dieser Stelle werden zwei Ebenen des Sicherheitsmanagements unterschieden. Zum einen die Integration der IPR-Information durch Anhängen von Textketten bzw. Signaturen in Form eines Attributobjektes (*Label Object*) an den Gruppenknoten des betreffenden Objektes (Ebene 1) . Zum anderen durch Einbettung der IPR-Information als Wasserzeichen in die Objektdaten selbst (Ebene 2).

Zunächst soll der Entwurf von Ebene 1 beschrieben werden. Beim Einlesen eines Dokumentes werden die IPR-Informationen aus der Datenbank extrahiert und in das interne Dokumentmodell integriert. Für 3D Dokumente wird dieses Dokumentmodell vom Szenengraph der Visualisierungsanwendung vorgegeben.

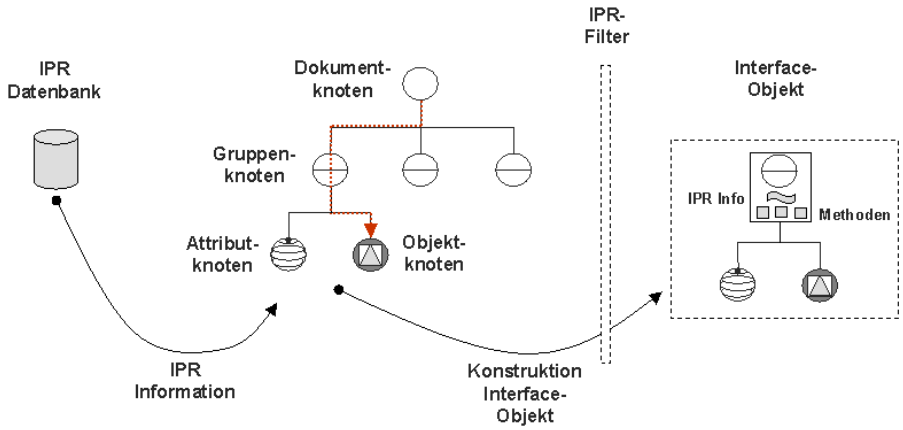

Abbildung 6.11: Integration durch Verwendung eines Attributknotens und durch Konstruktion eines Interface-Objektes.

In Abbildung 6.11 ist die Integration der IPR-Information in den Szenengraph dargestellt. Dabei wird ein Attribut-Knoten in die übergreifende Gruppe des betreffenden Objektes eingefügt. Dieser Label-Knoten enthält als String die oben angegebene IPR-Information.

Um die so integrierte Information im Falle der Selektion des Objektes durch einen Benutzer wirksam zu machen, wird das Interface-Objekt des Gruppenknotens in der Weise erzeugt, daß es als zusätzliche Daten die IPR-Information enthält und Methoden zum Sicherheitsmanagement bereitstellt. Dieses Vorgehen wird von dem in Abschnitt 6.3.2 vorgestellten Document-Request-

Broker-Ansatz unterstützt. Wenn das Objekt mit Hilfe der Client-Anwendung selektiert wird, steht dem Benutzer das zugehörige Interface-Objekt mit seinen Methoden zur Verfügung, um unter anderem Objektrepräsentationen zu übertragen oder die Metadaten abzufragen. Durch den Vergleich der IPR-Informationen und die Rechte des Benutzers, die mit seiner Aktion übertragen wurden, sind nur diejenigen Methoden zugänglich, die die Schutzrechte des Objektes nicht verletzen, z.B. die Methode *getOR(LoI≙Klasse B)* im Falle von Klasse B- Objekten. Dabei wird das Klassifizierungsschema (*Document-Distribution-Classes*) aus Abschnitt 2.4.4 verwendet, das regelt, auf welche Weise die Objektdaten verteilt werden dürfen. Auf diese Weise wird ein Filter (Abbildung 6.11: IPR-Filter) für Objektdaten realisiert, durch den der Zugriff der Benutzer kontrolliert werden kann.

Zum Vergleich der IPR-Informationen mit den Rechten eines Benutzers wird eine Signatur verwendet, die der Client bei der Registrierung am Server erhalten hat. Diese Signatur, die unter anderem die Benutzerrechte in codierter Form enthält, wird lokal vom Benutzer vorgehalten und mit jeder Aktion zum Server übertragen. Die Signatur des Clients ist ebenfalls am Server gespeichert und kann so zur Identifikation des Benutzers und zum Abgleich der Rechte herangezogen werden. Durch entsprechende Verschlüsselung der Signatur wird dieses Verfahren sicher gegen Manipulationen von außen.

Nun wird auf den Entwurf von Ebene 2 zur Einbettung von Wasserzeichen in 3D Objekte eingegangen. Diese Ebene des Sicherheitsmanagements baut auf dem vorher besprochenen Ansatz der Interface-Objekte auf. Genau wie in Ebene 1 werden die IPR-Informationen als Attributknoten in den Szenengraph integriert. Bei der Erzeugung des Interface-Objektes werden nun allerdings Methoden zur Verfügung gestellt, die eine Einbettung von Wasserzeichen in die Objektdaten ermöglichen. Diese Methoden werden automatisch ausgeführt, wenn ein Benutzer eine Objektrepräsentation erzeugen und übertragen möchte.

Das Interface-Objekt wird, wie oben beschrieben, als erweitertes Gruppenobjekt angelegt, indem es von einer OpenInventor Gruppen-Klasse (*SoSeparator*) abgeleitet wird und zusätzliche Daten und Methoden enthält. Es wird mit der Kennung der erlaubten IPR-Klasse als Konstruktor-Parameter am Server erzeugt und dem Client über eine CORBA Objektreferenz bekannt gemacht. Durch die IPR-Klasse und die Client-Signatur wird die Verfügbarkeit der Methoden des Objektes gesteuert. Auf diese Weise kann ein Benutzer nur mit den erlaubten Objektrepräsentationen arbeiten.

6.3.4 Die Komponenten der Bilddatenschicht

In diesem Abschnitt wird die Verteilung von Bilddaten in der SCA3D-Architektur anhand des Entwurfs der Bilddatenschicht beschrieben. Auf Server-Seite werden vom Offscreen-Rendering-Prozeß mit der eingestellten Server-Framerate Bilder erzeugt, die die Ergebnisse der Beleuchtungssimulation des 3D Dokumentes darstellen. Diese Bilddaten sollen auf Client-Seite auf die in der Szene enthaltene Projektionsfläche abgebildet werden. Dafür müssen sie an jede verbundene Client-Anwendung gesendet werden. Die Verteilung der Bilddaten soll jedoch ohne Abhängigkeit zwischen den einzelnen Übertragungsprozessen stattfinden, um zu verhindern, daß die langsamste Übertragung alle anderen Verbindungen bremst. Die Bilddaten werden über Socket-Verbindungen mit dem TCP/IP-Protokoll bzw. mit dem UDP-Protokoll gesendet. Es werden Unicast-Verbindungen verwendet, d.h. für jeden Teilnehmer werden die Bilddaten separat übertragen. Multicast-Verbindungen werden nicht verwendet, weil die Möglichkeit bestehen soll, jeden Benutzer mit einer unterschiedlichen Bilddatenrate zu versorgen.

Der Aufbau der Bilddatenschicht ist in Abbildung 6.12 gezeigt. Die Bilder werden von der Methode $Update()$ der Server-Anwendung in einen Bild-Ringbuffer geschrieben, der eine Ent-

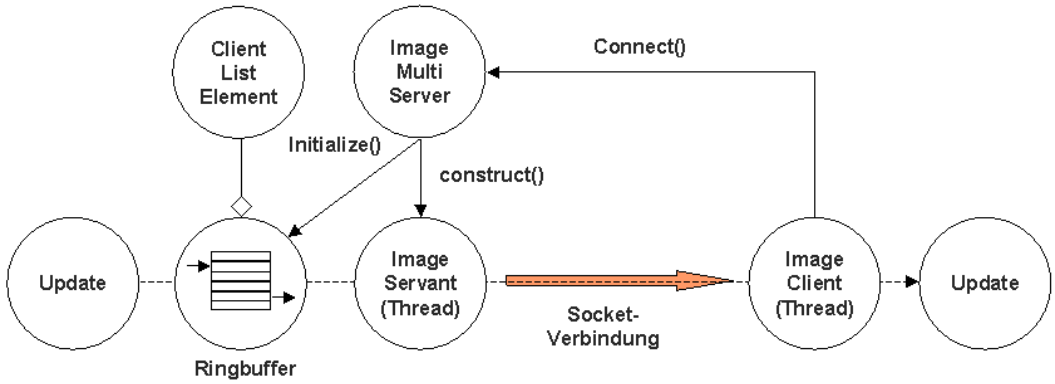

Abbildung 6.12: Aufbau der Bilddatenschicht der SCA3D-Architektur.

kopplung von Bilderzeugung und Bildübertragung für jeden Benutzer zuläßt.

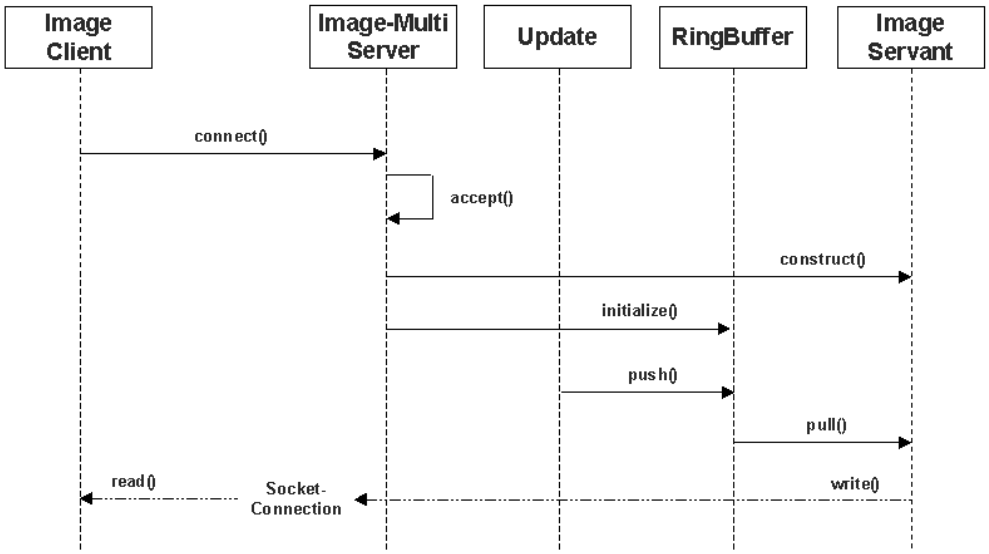

Abbildung 6.13: Sequenzdiagramm des Verbindungsaufbaus und des Bilddatenaustausches

Beim Anmelden eines Benutzers wird ein eigener Ringbuffer für diesen angelegt und in einer Instanz der Klasse $ClientListElement$, die für jeden Benutzer generiert wird, abgelegt. Ebenfalls auf Server-Seite liest eine Instanz der Klasse $ImageServant$, die von der $Thread$-Klasse abgeleitet ist und als eigenständiger Prozeß abläuft, die Bilder aus dem jeweiligen Bild-Buffer aus und schreibt sie auf die Socket-Verbindung zwischen Server und Client. Auf Client-Seite liest eine Instanz der Klasse $ImageClient$, die ebenfalls als eigenständiger *Thread* arbeitet, von dieser Socket-Verbindung die Bilddaten und schreibt sie in einen internen Speicherbereich. Von dort werden die Bilddaten von einer Instanz der Klasse $Update$ der Client-Anwendung als Textur auf die

Projektionsfläche abgebildet.

In Abbildung 6.13 ist das Sequenzdiagramm des Verbindungsaufbaus und des Zugriffs auf den Ringbuffer beim Bilddatenaustausch gezeigt. Bei der Anmeldung einer Client-Anwendung wird eine Socket-Instanz erzeugt, die eine Connect-Auforderung an den wartenden Server-Socket sendet. Der Server-Socket reagiert durch eine $accept()$-Methode und baut eine Socket-Verbindung mit der angegebenen Port-Nummer auf. Anschließend wird eine Instanz der Klasse $ImageServant$ als Thread auf Server-Seite erzeugt, die die neue Socket-Instanz als Eingabeparameter übernimmt. Parallel wird der Bild-Ringbuffer mit der geforderten Bildauflösung initialisiert. Nun können von der $Update$-Instanz Bilddaten mit der Server-Framerate in den Ringbuffer geschrieben werden, die wiederum von dem $ImageServant$-Thread entsprechend der an den Client angepaßten Bildrate ausgelesen werden. Auf einen Buffer-Überlauf bzw. ein leeren Buffer wird durch einen Ausnahme-Aufruf reagiert.

6.4 Zusammenfassung

In diesem Kapitel wurden die Schritte beim Entwurf einer Software-Architektur allgemein vorgestellt und die Vorteile der Verwendung der Sprache UML im Entwicklungsprozeß verdeutlicht. Für die Software-Architektur SCA3D wurden die Anforderungen an das System mit Hilfe von UML-Diagrammen beschrieben und Sichten auf das System aus Benutzerperspektive gezeigt. Daran anschließend wurde die Designphase des Entwicklungsprozesses ebenfalls mit UML-Diagrammen erläutert. Dabei standen die Visualisierungskomponenten, der Dokument-Request-Broker, die Level-of-Information-Komponenten und die Bilddatenschicht im Vordergrund. Im folgenden Kapitel wird nun die Implementierungsphase des Entwicklungsprozesses beschrieben und die Randbedingungen der Hardware- und Software-Umgebung in den Entwurf miteinbezogen. Am Ende des Kapitels wird die Software-Architektur SCA3D als ein verteiltes objekt-orientiertes System mit einer Software-technischen Realisierung in einer Hardware-Umgebung beschrieben sein.

Kapitel 7

Implementierung der Architektur SCA3D

In diesem Kapitel wird die Implementierung der Software-Architektur für skalierbare Szenen beschrieben. Die verwendeten Werkzeuge, die erstellten und benutzten Hard- und Software-Komponenten und die Kommunikation zwischen den Komponenten werden im Detail dargestellt. Da im Vorfeld der Entwicklung des SCA3D-Konzeptes eine Vorgängerarchitektur erarbeitet wurde, die den Zugriff auf server-seitige 3D Modelle über Videobjektrepräsentationen erlaubt, wird diese als Implementierung eines ersten Prototypen kurz beschrieben. Daran anschließend wird die Implementierung der SCA3D-Architektur als objekt-orientiertes verteiltes System erläutert.

7.1 Einführung

Die Software-Architektur SCA3D soll möglichst plattform-unabhängig realisiert werden. Für verteilte Systeme bietet sich hierfür der CORBA-Standard zur objekt-orientierten Entwicklung an, da er, wie in Kapitel 3.3 beschrieben, Technologietransparenz ermöglicht. Dadurch können auch Komponenten, die in unterschiedlichen Programmiersprachen entwickelt werden, in einem verteilten System zusammenarbeiten. Zur Realisierung von SCA3D wird die CORBA-Implementierung JavaIDL des Java-Developement-Kits verwendet. Die Software-Komponenten sind in Java geschrieben und laufen damit ohne größere Anpassungen auf allen Plattformen, für die eine *Java-Virtual-Machine* existiert. Die Visualisierungskomponenten in SCA3D sind mit dem OpenInventor API umgesetzt worden. Die eigentlichen Klassen des Visualisierungs-API sind in C++ geschrieben und wurden mit Hilfe eines *Java-Native-Interfaces* [Lia99] in die Java- Klassen eingebunden. Dadurch wird eine höhere Geschwindigkeit bei der Visualisierung ermöglicht und die Vorteile der Java-Programmierung werden gleichzeitig ausgenutzt. Allerdings wird hierdurch eine eigene Runtime-Umgebung für das OpenInventor-API auf jeder Plattform benötigt und die Plattformunabhängigkeit wird zum Teil aufgegeben. Alternativen sind die Verwendung einer vollständig in Java implementierten OpenInventor-Klassenbibliothek (z.B. *3D-MasterSuite Java* der Firma TGS) oder der Java-eigenen Visualisierungsbibliothek *Java3D* (siehe z.B. [SRD98]).

7.2 Verwendung von Software- und Hardware-Werkzeugen

Die Software-Architektur SCA3D besteht aus mehreren Schichten, die mit Hilfe verschiedener Werkzeuge umgesetzt werden. Dies sind die HTTP-Kontrollschicht, die Visualisierungsschicht,

die Dokumentkontrollschicht und die Bilddatenschicht. Die verwendeten Software- und Hardware-Tools zur Realisierung der Architektur SCA3D werden in diesem Abschnitt beschrieben.

7.2.1 3D Visualisierung: Das Radiance-System und die OpenInventor-Bibliothek

Für die 3D Visualisierung bzw. das Rendering von 3D Modellen wurden im Laufe der Arbeiten zwei unterschiedliche Werkzeuge verwendet. Anfänglich wurde die Raytracing-Software *Radiance* [War94] benutzt, die mit einem relativ hohen Zeitaufwand physikalisch korrekt berechnete Beleuchtungssimulationen erzeugen kann. Für die Entwicklung der SCA3D-Architektur wurde dann im folgenden die szenengraph-basierte Visualisierungsbibliothek OpenInventor [Wer98] benutzt, die zwar nur die Möglichkeit eines lokalen Beleuchtungsmodelles bietet, dafür aber mit interaktiven Bildwiederholraten aufwarten kann. Beide Werkzeuge werden nun näher beschrieben.

- *Radiance*: Da der Raytracing-Algorithmus von Radiance Spiegelungen, Transmissionen und diffuse Reflexion durch Ausbreitung von virtuellen Lichtstrahlen nachbildet, wirken die erzeugten Bilder des 3D Modells sehr realistisch. Allerdings wird dafür in Kauf genommen, daß keine wirklich interaktive Visualisierung möglich ist. Trotzdem bietet die modifizierte Raytracing-Software Möglichkeiten zum Erzeugen von Objektmasken, sodaß Bildobjekte mit Konturinformation als Repräsentation zum Client übertragen werden können, die dort als Objektreferenz dienen. Die nötigen Modifizierungsschritte wurden für die Radiance-Software durchgeführt und damit die Codierung von Videoobjekten als Repräsentation von 3D Objekten in einer verteilten Umgebung mit Hilfe eines MPEG-4 Videocodecs ermöglicht.

  Radiance stellt keine Visualisierungsbibliothek (API) zur Erweiterung anderer Programme mit 3D Visualisierungsfunktionalität dar, sondern ist als eigenständiges Rendering-System zum Einsatz in den Bereichen Lichtdesign und Architektur entwickelt worden. Der Programmcode ist nicht objekt-orientiert und wurde in der Programmiersprache C entwickelt. Radiance verwendet rekursives Raytracing mit einigen Erweiterungen, um die Beleuchtungssimulation mit einem optimalen Verhältnis von Genauigkeit und Geschwindigkeit zu berechnen. Da der Quellcode frei zugänglich und gut dokumentiert ist, konnte das System erfolgreich an die Aufgaben in der Praxis und im wissenschaftlichen Umfeld angepaßt werden. Dadurch ist es heute ein weitverbreitetes Werkzeug zur Erzeugung von physikalisch realistischen Bildern von 3D Oberflächenmodellen und zur numerischen Auswertung von Beleuchtungssituationen in virtuellen Welten.

- *OpenInventor*: Zur Entwicklung verteilter Umgebungen, die 3D Visualisierung für mehrere Benutzer unterstützen soll, muß die verwendete Visualisierungsbibliothek interaktive Frameraten, d.h. möglichst eine Bildwiederholrate von 25 bis 30 Frames pro Sekunde, erlauben. Denn der Eindruck eines kooperativen Arbeitsprozesses in verteilten Umgebungen kann nur durch interaktive Frameraten bei der 3D Visualisierung aufrechterhalten werden. Jeder Teilnehmer einer Visualisierungskonferenz sollte die Aktionen der anderen Teilnehmer mit einer minimalen Verzögerungszeit wahrnehmen können, die eine obere Schranke nicht überschreitet. Um solche Frameraten zu erreichen, kann ein Rendering-System mit einem globalem Beleuchtungsmodell nicht verwendet werden. Daher arbeiten Visualisierungs-APIs, wie die von der Firma SGI entwickelte OpenInventor-Bibliothek, mit einem lokalen Beleuchtungsmodell, bei dem unter anderem keine Schatten oder Reflexionen berechnet werden. Zur Beleuchtungssimulation wird die in Abschnitt 3.2.2 durch Formel 3.5 angegebene Gleichung gelöst, d.h. OpenInventor benutzt als Standard das Phong-Beleuchtungsmodell.

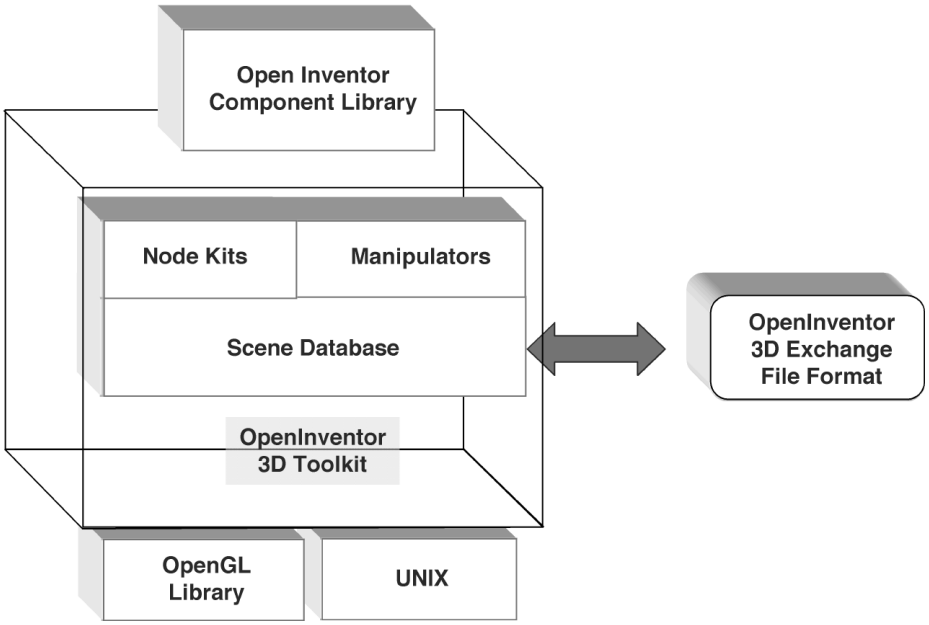

Abbildung 7.1: Die OpenInventor-Architektur nach [Wer98]

OpenInventor stellt eine Bibliothek von objekt-orientierten Klassen zur Verfügung, die es einem Anwendungsprogrammierer ermöglicht, interaktive Graphikanwendungen zu entwickeln. Das OpenInventor-System ist in C++ geschrieben, bietet aber auch die Möglichkeit zur Entwicklung in der Sprache C. Durch Verwendung eines Java-Native-Interface können außerdem Graphikprogramme mit OpenInventor vollständig in Java entwickelt werden. Zur Zeit sind die bekanntesten Schnittstellen zwischen OpenInventor und Java die *3D MasterSuite for Java* der Firma TGS und die frei verfügbare Schnittstelle *Kahlua* [WK96], die von an der Brown University (USA) entwickelt wurde und nun vom Institut für Computergraphik der Fraunhofer-Gesellschaft weiterbetreut wird. Die Basisarchitektur des OpenInventor-Systems ist in Abbildung 7.1 dargestellt.

Zur Verdeutlichung der Funktionsweise des OpenInventor-API soll hier das Konzept der internen Szenendatenbank und der Anwendung von Aktionen auf den internen Szenengraphen näher erläutert werden. Die Basiseinheit der von OpenInventor verwendeten Szenendatenbank sind die Knoten, die zum Aufbau einer 3D Szene benutzt werden. Eine als Baumstruktur geordnete Zusammenstellung von Knoten wird als Szenengraph bezeichnet und stellt einen gerichteten, azyklischen Graph dar. Bei einem gerichteten, azyklischen Graph wird für die Baumstruktur eine Richtung vorgegeben, mit der man von einem Knoten über eine Verbindung zum nächsten Knoten gelangt. Außerdem beginnt und endet kein Pfad im Graph mit demselben Knoten, d.h. es existieren keine geschlossenen Pfade. Wird eine Aktion auf den OpenInventor-Szenengraph angewendet, so wird er von oben nach unten und von links nach rechts durchgegangen und die Aktion auf alle Knoten angewendet.

Die Szenendatenbank speichert und verwaltet einen oder auch mehrere Szenengraphen. Die Datenbankentitäten bzw. -primitive stellen Form-Knoten (*Shape Nodes*) (z.B. Kugel, Cone, Zylinder, Quad-Mesh), Eigenschaftsknoten (*Property Nodes*) (z.B. Material, Beleuchtungs-

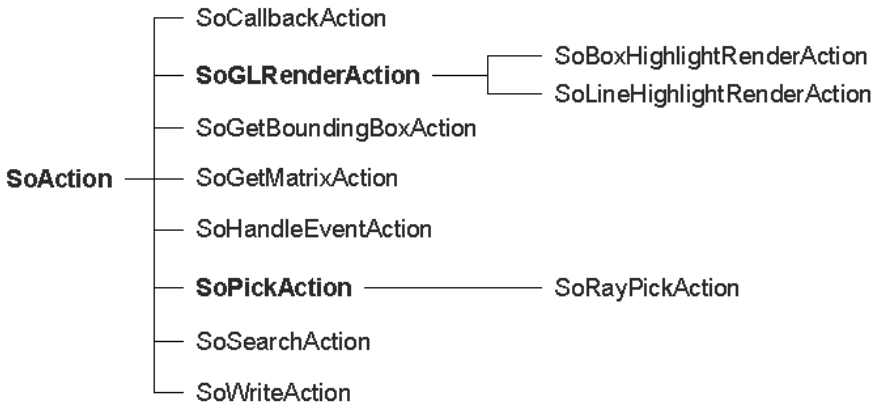

Abbildung 7.2: Hierarchie der SoAction-Klassen von OpenInventor

modell, Texturen) und Gruppen-Knoten (*Group Nodes*) (z.B. Separator, Level-of-Detail, Schalter) dar. Zusätzlich kann ein Szenengraph in der Datenbank auch Sensoren (*Sensors*) und sogenannte Maschinen (*Engines*) enthalten, mit denen Knoten verknüpft, Zustandsänderungen detektiert und dynamische Szenen erstellt werden können. Wenn eine Aktion angewendet wird (z.B. *SoGLRenderAction*, *SoGetBoundingBoXAction*, *SoWriteAction*), wird von der Datenbank der aktuelle Zustand (*Traversal State*) in einem eigenen Objekt verwaltet und die Veränderung der Zustandsparameter durch die Knoten protokolliert. Die Änderung des Traversal-States durch die Knoten beim Durchgang des Szenengarphen führt im Ergebnis zu der interaktiven Visualisierung, die vom Benutzer wahrgenommen wird. Jeder Knoten implementiert dabei ein Aktionsverhalten, das durch seine Klasse definiert wird. So wird das Verhalten im Falle einer *SoGLRenderAction*, die zur Ausgabe von gerenderten Bildern der Szenenobjekte führt, von einem Gruppenknoten als Traversierung der Unterknoten und anschließender Wiederherstellung der vorherigen Zustandes implementiert. Ein Material-Knoten ändert dagegen die Zustandswerte für das Material im Traversal-State. Wenn eine Formknoten erreicht wird, wird dessen Geometrie mit den Parametern des Traversal-State gerendert und die Bilddaten ausgegeben bzw. gespeichert. Aktionen können auf Knoten, Pfade und Listen von Pfaden angewendet werden.

OpenInventor bietet eine Austauschformat für Szenen, das den Aufbau des Szenengraphen mit einer festgelegten Syntax wiedergibt. Dieses Format kann als Ausgabeformat des Datenmodells des Szenendatenbank angesehen werden. Beim Einlesen einer solchen OpenInventor-Datei (Suffix: *.iv), wird diese Information auf die interne Datenbank der Anwendung abgebildet und somit die 3D Szene für die interaktive Visualisierung zugänglich gemacht. Eine Szene kann in eine Ausgabedatei geschrieben werden, indem ein *SoWriteAction* auf ihren Pfad angewendet wird. Aus dem OpenInventor-Austauschformat wurde auch das VRML-Format für 3D Visualisierungsanwendungen im WWW-Umfeld entwickelt.

7.2.2 Hardware-Komponenten

In diesem Abschnitt wird die Entwicklungs- und Testumgebung der Software-Architektur SCA3D vorgestellt. Dabei werden die verwendeten Rechner mit ihrer Hardware, den Betriebssystemen und der jeweiligen Netzanbindung beschrieben. Alle Komponenten der Architektur arbeiten auf Unix- und Windows-Systemen, da zur Entwicklung die plattform-unabhängige Sprache Java verwendet wird. Allerdings verlangt das OpenInventor-API eine eigene Runtime-Version auf jedem Betriebssystem.

Da die Entwicklungsumgebung mit Runtime-Lizenzen für Windows-, SGI-Irix- und Sun-Solaris-Betriebssysteme ausgestattet wurde, können alle Komponenten beliebig verteilt werden. Allerdings wurde im Verlauf der Entwicklung eine Verteilung gewählt, bei der ein Server auf Unix-Maschinen (Unix-Workstation: Sun-Solaris) und die Client-Anwendungen hauptsächlich auf Windows-Rechnern (PC: WindowsNT, Notebook: Windows98) betrieben wurden. Einige Tests wurden auch mit spezialisierten Graphikmaschinen von SGI als Server vorgenommen. Die Anpassung auf spezialisierte Graphik-Hardware wurde allerdings nicht vertieft, um die Einsatzmöglichkeiten der Software-Architektur nicht schon in der Entwicklungsphase einzuschränken. In Tabelle 7.3 ist eine Zusammenstellung der verwendeten Maschinen in der Entwicklungsgumgebung gezeigt.

Rechner	Betriebssystem	Netzanbindung	Komponenten
Sun Enterprise 450	Solaris 2.7	100 Mbit Ethernet	SCA3D Server
2x 300 Mhz UltraSparc II		altern. 0.5 - 622 Mbit ATM	
Sun Ultra 30	Solaris 2.7	100 Mbit Ethernet	SCA3D Client
300 Mhz UltraSparc II		altern. 0.5 - 622 Mbit ATM	
Dell PC			
Pentium II 200 MHz	WindowsNT	100 Mbit Ethernet	SCA3D Client
Sony Notebook Vaio	Windows 98	100 Mbit Ethernet	SCA3D Client
Pentium III 600 Mhz		altern. 56 kBit Modem	

Abbildung 7.3: Beschreibung der Hardware der Entwicklungsumgebung.

Die Entwicklung und die meisten Tests wurden in dem LAN (*Local Area Network*) des Instituts für Medienkommunikation der GMD durchgeführt. Alle oben aufgeführten Rechner waren über ein Ethernet mit einer Bandbreite von maximal 100MBit/s vernetzt. Der verwendete mobile PC (*Notebook*) konnte alternativ über ein Modem mit 56kBit/s angeschlossen werden. Zusätzlich zum Ethernet-LAN war ein ATM-Netz mit maximal 622 MBit/s Bandbreite verfügbar. Die beiden oben aufgeführten Unix-Maschinen waren sowohl an das Ethernet, als auch an das ATM-Netz angeschlossen. Mit Hilfe der ATM-Technologie konnte eine Verbindung zwischen SCA3D-Client und SCA3D-Server aufgebaut werden, bei der die Bandbreite regulierbar war. Der Bereich, in dem die Bandbreite variiert wurde, lag zwischen 128 kBit/s und 100 MBit/s. Eine ATM-Verbindung zeichnet sich weiterhin dadurch aus, daß die angeforderte Bandbreite auch garantiert werden kann, d.h. vollständig für die Client-Server-Kommunikation zur Verfügung steht. Im Ethernet bzw. im heterogenen Internet ist dies nicht der Fall, da es sich um *Best-Effort*-Verbindungen handelt.

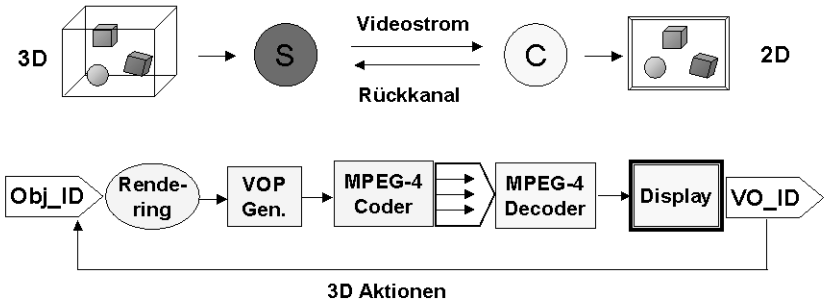

Abbildung 7.4: Zugriff auf server-seitige 3D Objekte über die Kennung von Videoobjekten

7.3 Realisierung des ersten Prototypen: Remote-Rendering über MPEG-4 Videoströme

Zunächst soll eine reine Remote-Rendering Architektur vorgestellt werden, bei der die 3D Geometrie während der Visualisierung vollständig auf dem Server verbleibt. Hierzu wird ein server-seitiges Visualisierungsmodul entworfen, das 3D Geometrie einliest und daraus Bilder durch Rendering erzeugt. Diese Bildinformation wird an die Client-Rechner übertragen und dort mit Hilfe eines Displays angezeigt. Der Benutzer hat die Möglichkeit über einen Rückkanal zum Server die 3D Visualisierung dort zu steuern, indem er zum Beispiel die Kameraposition oder die Bildparameter ändert. Manipulation der 3D Objekte selbst erfordert eine Möglichkeit zur Referenzierung dieser Objekte anhand ihrer Objektkennung durch die Client-Anwendung. Die dazu benötigte zusätzliche Information kann, wie im folgenden beschrieben, durch Verwendung von objekt-basierter Videocodierung bei der Bildübertragung geliefert werden. Ein objekt-basierter Videocodec erfüllt hier zwei Funktionen, indem die Datenmenge reduziert wird und gleichzeitig Objektinformation erhalten werden kann. Bei dieser Architektur wird das visuelle Feedback für die Interaktion nur über die Bildinformation von Server gegeben, daher müssen Probleme durch unvermeidliche, zeitliche Verzögerungseffekte bei der Netzübertragung berücksichtigt werden. Die Ergebnisse, die mit diesem Ansatz erzielt wurden, sind in [Loe00a] beschrieben. Zur Steuerung dieser verteilten Visualisierungsanwendung wird ein Protokoll zur Rückübertragung von Kontrollbefehlen benötigt, das mit einer Schnittstelle der server-seitigen Graphikanwendung kommuniziert. Auch die Bildparameter der Codierung werden über dieses Protokoll kontrolliert.

Beim *Visual-Interaction-Approach* (VIA) [Loe00a] zur verteilten 3D Visualisierung wird ein 3D Modell auf dem Server-Rechner gerendert und ein visuelles Feedback in Form von codierten Bildern zum Client übertragen. Dieser kann mit Hilfe eines Rückkanals die Visualisierungsanwendung am Server steuern, indem er z.B. den Blickpunkt oder -winkel ändert oder die Szene selbst manipuliert. Der Zugriff auf die 3D Objekte am Server erfolgt in diesem Fall über die im MPEG-4 Videostrom mitgelieferte Kennung der Videoobjekte. Ein Objekt kann am Client für die Interaktion selektiert werden, so daß es als Videoobjekt in einen separaten Elementarstrom codiert wird. Hierdurch ist am Client die Kontur- , Textur- und Bewegungsinformation eines selektierten Objektes verfügbar. Der Benutzer kann nun eine 3D Aktion festlegen, die mit der Objektkennung und den Transformationsparametern zum Server übertragen wird und dort auf das 3D Modell angewendet wird. Durch diese Methode kann ein Benutzer mit der 3D Geometrie interagieren ohne die Geome-

triedaten selbst auf seinen Rechner übertragen zu müssen. Alternativ zu einem selektierten Objekt können auch alle sichtbaren Objekte einer Szene dynamisch in Elementarströme einer MPEG-4 Videoszene codiert werden. Der MPEG-4 Standard erlaubt maximal 1024 Objekte in einer Szene, so daß Interaktion mit den Objekten des sichtbare Bereich eines 3D Modells mit dieser Methode mit einer guten Auflösung realisiert werden kann. In Abbildung 7.4 ist die Übertragung von MPEG-4 Elementarströmen und die Referenzierung von 3D Objekten mit Hilfe der Kennungen der Videoobjekte am Client veranschaulicht.

Die wesentlichen Komponenten für diesen Ansatz sind auf der Server-Seite ein 3D Renderer, ein MPEG-4 Videocoder und eine Server-Anwendung zur Übertragung der Videoströme und zum Empfang der Benutzeraktionen. Auf Client-Seite wird ein MPEG-4 Videodecoder und eine interaktive Display-Anwendung benutzt, um die Videoströme zu decodieren, in der richtigen Weise zusammenzusetzen und um die Client-Interaktion mit der Szene zu ermöglichen. Diese Identifizierung von Objekten erfolgt bei der Komposition des Gesamtbildes, indem die Shape-Information mit der aktuellen Zeigerposition und dem Modus der Mauseingabe verglichen wird. Aus den erhaltenen Aktionsparametern wird eine Aktionsnachricht zusammengebaut, die neben der Objekt- und Benutzerkennung auch die Frame-Nummer, die Transformationskennung und eine 3D Transformationsmatrix enthält. Diese Aktionsnachrichten werden über einen Rückkanal zum Server übertragen und dort ausgewertet, bevor sie an die Visualisierungsanwendung weitergegeben werden. Zur zeitlichen Synchronisation wird die mitgelieferte Frame-Nummer der Nachricht verwendet, so daß zeitliche Verzögerungen im Netz berücksichtigt werden können, auch wenn mehrere Benutzer an einer Szene arbeiten.

7.4 Implementierung der SCA3D-Architektur für den Level-of-Information-Ansatz

Aufbauend auf den Ergebnissen aus der Umsetzung des ersten Konzeptes wird die Software-Architektur SCA3D für skalierbare Szenen implementiert, wobei Remote-Visualisierung und lokale Visualisierung kombiniert werden. Zur 3D Visualisierung wird die OpenInventor Software benutzt, da sie für interaktive Visualisierungsanwendungen aufgrund der erreichbaren Bildwiederholraten besser geeignet ist. Die Umsetzung erfolgt in der objekt-orientierten Programmiersprache Java und zur Entwicklung wird die in Abschnitt 7.2.2 angegebene Hardware-Umgebung genutzt.

7.4.1 Das Schichtenmodell der SCA3D-Architektur

Um den Aufbau der Architektur zu verdeutlichen wird nun ein Schichtenmodell vorgestellt, das die verschiedenen Ebenen mit den darin verwendeten Technologien darstellt. Die vier Hauptschichten der SCA3D-Architektur sind, wie in den vorhergehenden Kapiteln schon angedeutet, die Web-Schicht, die 3D Visualisierungsschicht, die Dokumentkontrollschicht und die Bilddatenschicht. In Abbildung 7.5 ist dieses Schichtenmodell gezeigt.

Im folgenden werden die Schichten und ihre interne Umsetzung näher beschrieben:

- *WWW-Schicht*:

  Die Web-Schicht wird benutzt, um dem Benutzer oder einem Administrator den Zugriff auf den SCA3D-Server und die server-seitige Datenbank zu ermöglichen. Es werden HTML-Formulare und server-seitige CGI-Skripte zur Kontrolle des Servers und zum Zugriff auf

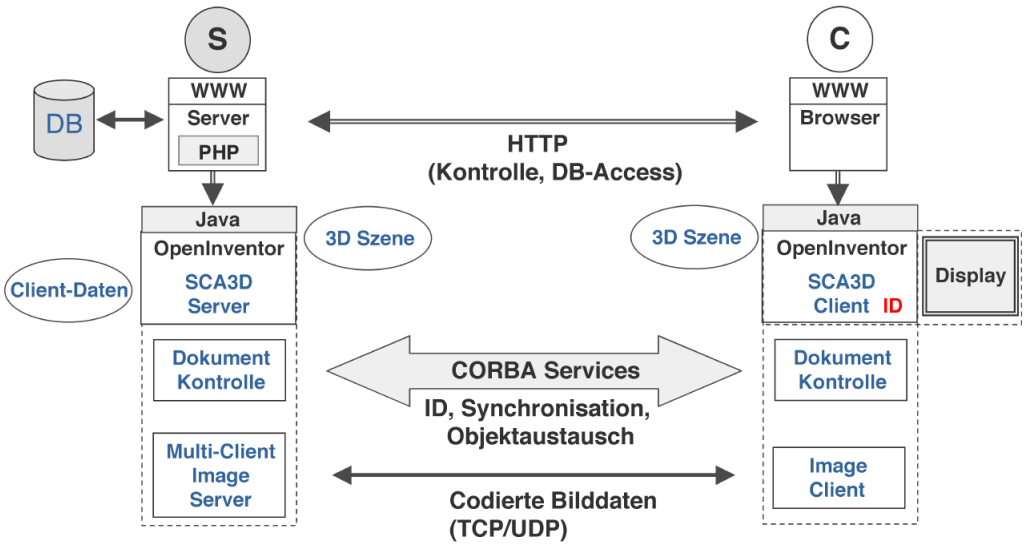

Abbildung 7.5: Aufbau der SCA3D-Architektur dargestellt als Schichtenmodell.

die Datenbank verwendet. Der Benutzer lädt zu Beginn die HTML-Start-Seite des SCA3D-Servers, indem er die entsprechende URL in seinem Browser anwählt. Ein Ausschnitt der Start-Seite ist in Abbildung 7.6 gezeigt. Dort sind die Formular-Buttons zum Start bzw. zum Beenden des Server-Prozesses und des CORBA-Naming-Servers zu erkennen. Der Benutzer kann sich durch Auslösen des Server-Status-Buttons den aktuellen Zustand des Servers, also die laufenden Prozesse und offene Netzverbindungen im Browser anzeigen lassen. Das 3D Dokument, das beim Start geladen wird, ist als Bild über dem Start-Button des SCA3D-Servers gezeigt. In diesem Fall handelt es sich um ein dreidimensionales Gebäudemodell im OpenInventor-Format.

Bevor eine Visualisierungssitzung mit SCA3D stattfinden kann, muß der Naming-Server der Middleware (CORBA) entweder von einem Benutzer oder von einem Administrator gestartet werden. Falls der Server immer zugreifbar sein soll, wird der Naming-Server auf der Server-Maschine vom Administrator gestartet und der SCA3D-Server im Anschluß daran aufgerufen. Dann können sich die Benutzer beliebig an- und abmelden, während der Server-Prozeß und der Naming-Server ständig weiterlaufen. Alternativ kann auch der erste Benutzer den Server starten und alle weiteren Benutzer müssen dann lediglich ihren SCA3D-Client starten. In diesem Szenario kann der erste Benutzer sich zu jedem Zeitpunkt abmelden, ohne daß die Visualisierungssitzung der anderen Benutzer gestört wird. Ein Skript auf Server-Seite kann dann dafür sorgen, daß beim Abmelden des letzten Benutzers der Server-Prozeß und der Naming-Server beendet werden.

- *3D Visualisierungsschicht*:

  Diese Schicht dient zur interaktiven Visualisierung der 3D Dokumente, die in den Server geladen wurden und zum Client verteilt werden. Dafür wird auf Server-Seite und auf Client-Seite ein Szenengraph aufgebaut und das OpenInventor-API zum Rendering und zur Interaktion mit den 3D Daten verwendet. Diese Schicht wird durch eine Gruppe

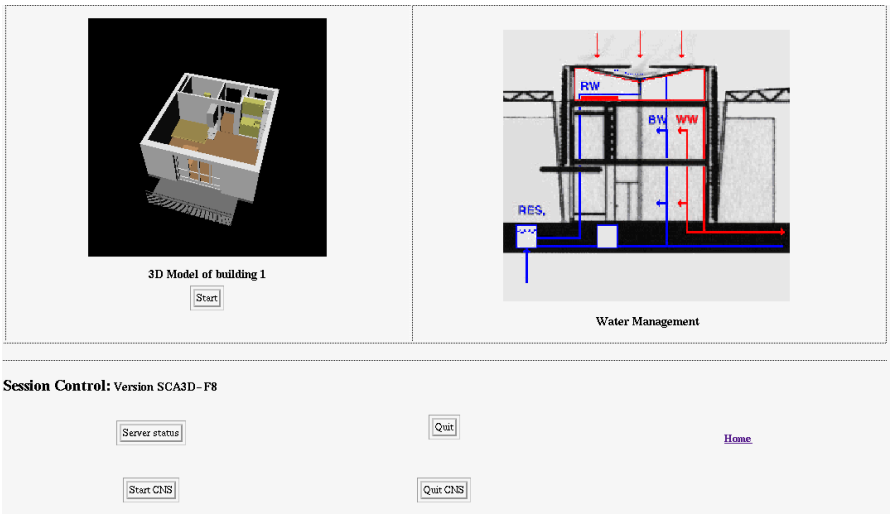

Abbildung 7.6: Die GUI-Oberfläche der WWW-Schicht. Quelle der Einzelbilder und Modelle: ETH Zürich, Fachbereich Architektur.

von Java-Klassen gebildet, die über das Java-Native-Interface (JNI) auf die OpenInventor-Klassenbibliothek zugreifen, die in C++ programmiert ist. Die Schnittstelle zwischen Java und OpenInventor ist in [WK96] genauer beschrieben. Die Klassenbibliothek mit dem Namen *Kahlua* ist zum Zeitpunkt der Veröffentlichung dieser Arbeit unter der URL *http://www.igd.fhg.de/CP/kahlua/index.html* verfügbar.

Die Techniken zur 3D Visualisierung und verteilten Synchronisation werden in der Visualisierungsschicht der SCA3D-Architektur in der gleichen Weise angewendet, wie in Abschnitt 4.2 beschrieben. Zu den Komponenten dieser Schicht gehört auch das Display zur Anzeige der gerenderten 3D Dokumente auf Client-Seite. Das Display und die Bedienelemente zur Interaktion sind im 3DVis-GUI der Client-Anwendung integriert. In Abbildung 7.7 sind die graphischen Oberflächen der Client-Anwendung 3DVis-GUI unf LoI-Panel als Beispiel gezeigt, bei dem zwei Benutzer mit einem CAD-Modell eines Gebäudemodells arbeiten. Auf Server-Seite wird eine Liste mit Benutzern und ihren anwendungsbezogenen Daten verwaltet, die ebenfalls zur Visualisierungsschicht gezählt wird, allerdings auch stark von der darunterliegenden Dokumentkontrollschicht benutzt wird, die nun beschrieben werden soll.

- *Dokumentkontrollschicht*:

  Diese Schicht dient zur Kontrolle des zwischen Server und Client verteilten 3D Dokumentes. Dazu werden für aktive Objekte Verbindungen zwischen dem Originalobjekt auf Server-Seite und der Objektrepräsentation auf Client-Seite über die in den vorigen Kapitel eingeführten Interface-Objekte hergestellt. Die Objektrepräsentation werden vom Server als Pfadinformation im OpenInventor-Austauschformat in eine Stringvariable geschrieben und dann als Parameter eines RPC an die Client-Anwendung gesendet. Dort werden sie in die Client-Szene eingefügt.

  Auf Client-Seite werden mittels der Performanzmessungen und der Einstellungen des Level-of-Information-Panel die LoI-Daten erzeugt, die über einen RPC an den Server übergeben

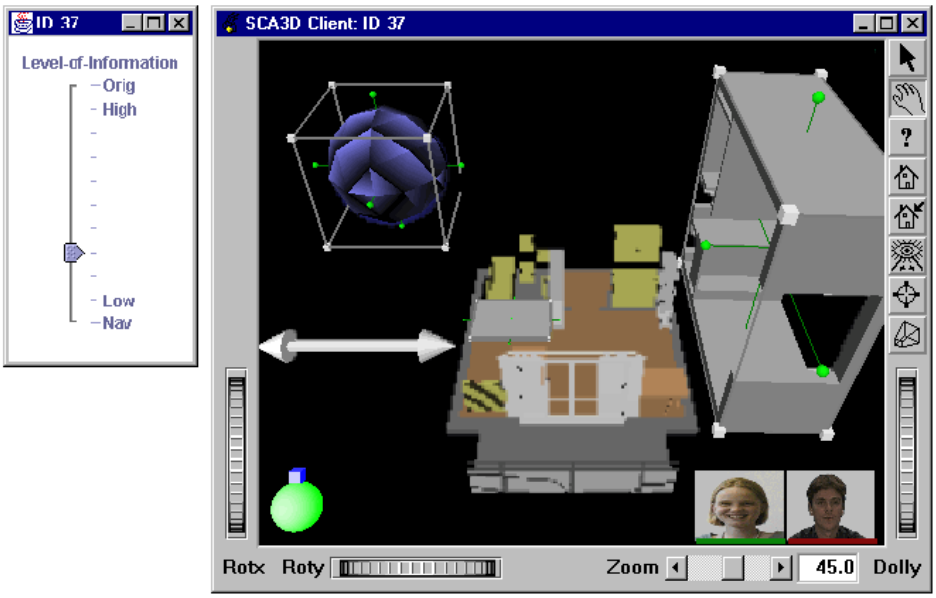

Abbildung 7.7: Die graphische Oberfläche 3DVis-GUI der SCA3D Client-Anwendung. Quelle des CAD-Modells: ETH Zürich, Fachbereich Architektur.

werden und dort bestimmen, welche Objektrepräsentationen für einen Benutzer generiert werden. Für jeden Benutzer wird auf Server-Seite eine Liste mit den Interface-Objekten und auf Client-Seite eine Liste mit den entsprechenden Objektrepräsentationen verwaltet. Die Interface-Objekte werden von der SceneControl-Servant-Klasse erzeugt und über diese auch zugegriffen. In Abbildung 7.8 sind der Aufbau und die Programmkomponenten zum Einsatz der Dokumentkontrollschicht dargestellt.

Zur Realisierung dieser Funktionen wird die CORBA-Implementierung JavaIDL der Firma Sun benutzt, die mit der Java-Umgebung frei erhältlich ist. Das SceneControl-Interface definiert in der CORBA-eigenen Interface-Sprache IDL die Methodenschnittstelle zum Zugriff auf die Dokumentkontrollobjekte der Software-Architektur. Eine *Listener*-Klasse auf Server-Seite und eine *Speaker*-Klasse auf Client-Seite sorgen für die grundlegende Kommunikation der SCA3D-Komponenten mit der CORBA-Schicht, die die Middleware-Funktionalität und die CORBA-Dienste zur Verfügung stellt. In diesen Klassen werden Verbindungen zum jeweiligen ORB (Object-Request-Broker) aufgebaut und Objektreferenzen für die remotezugreifbaren Objekte erzeugt. Zum Einsatz der Server- und Client-Komponente der SCA3D-Architektur wird zu Beginn ein Name-Server-Prozeß auf einem Rechner in der vernetzten Umgebung gestartet. In der CORBA-Implementierung von Java ist der Name-Server *tnameserv.exe* enthalten. Den SCA3D-Komponenten wird beim Start mitgeteilt, auf welchem *Host*-Rechner und über welchen Port der Name-Service zugänglich ist. Dadurch müssen Server und Client die Adressen gegenseitig nicht kennen und es wird die geforderte Ortstransparenz in der verteilten Architektur realisiert.

- *Bilddatenschicht*:

  Der Aufbau der Bilddatenschicht wurde im vorhergehenden Kapitel schon näher beschrieben.

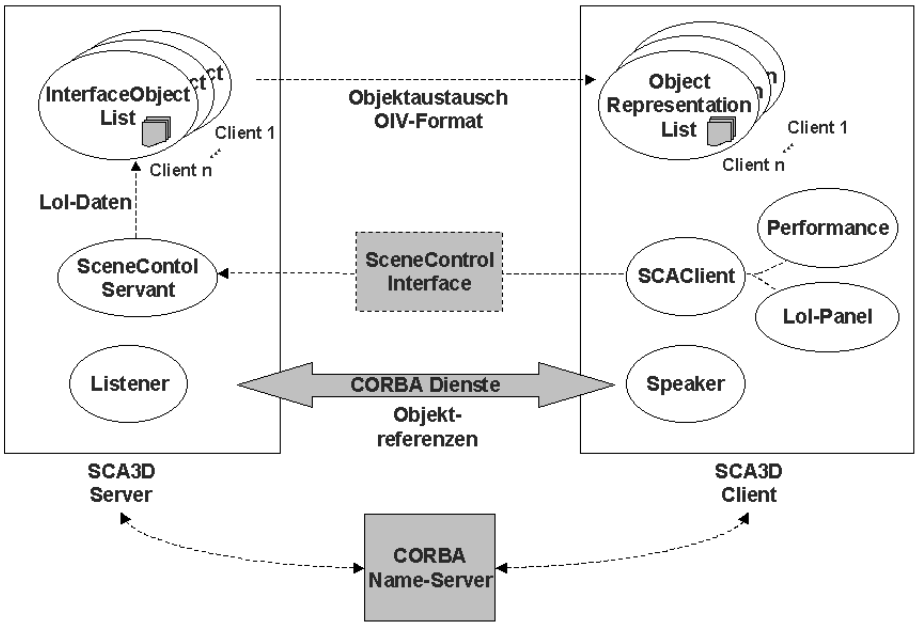

Abbildung 7.8: Aufbau und Einsatz der Dokumentkontrollschicht.

Daher wird hier nur auf die zugrundeliegende Technologie zur Übertragung eingegangen. Die in der SCA3D-Architektur verteilten Bildaten werden über Socket-Verbindungen vom Server zum Client übertragen. Dazu werden die von Java bereitgestellten Bibliotheksklassen *Socket* und *ServerSocket* verwendet. Die Bildaten werden von der OpenInventor *OffScreen-Renderer*-Klasse im RGB-Format mit einer einstellbaren Auflösung erzeugt und in einen Bildspeicher geschrieben. Aus diesem Buffer werden die Daten in einen Ring-Buffer geschrieben, wie es im vorhergehenden Kapitel beschrieben wurde. Die Bilddatenrate kann über einen OpenInventor *TimerSensor* eingestellt werden und wird entsprechend den Anforderungen der Client-Anwendungen geregelt.

Die Bildaten werden dann von einem unabhängigen Thread-Prozeß aus dem Ring-Buffer gelesen und über die Socket-Verbindung zum Client geschrieben. Bevor eine Socket-Verbindung aufgebaut werden kann, muß der Client sich beim der ServerSocket-Instanz, die auf Benutzeranfragen wartet, über einen vorgegebenen Port melden. Dann wird von der ServerSocket-Instanz ein mittels ihrer *accept*-Methode eine eigene Socket-Instanz erzeugt, über die Server- und Client-Anwendung die Bildaten austauschen. Damit sich ein Client mit dem Server verbinden kann, muß der Client-Anwendung die *Host*-Adresse und der Port bekannt sein, auf dem der Server horcht. Daher wird dem Bild-Client der SCA3D-Software-Architektur diese Information explizit beim Start mitgeteilt. Da die *Host*-Adresse in den meisten Fällen mit der Adresse des Rechners übereinstimmt, auf dem auch der CORBA-Name-Server läuft, reicht es aus diese Adresse einmal am Anfang zu übergeben. Um eine Ortstransparenz für die Bilddatenschicht zu erreichen, müßten die Bilddaten ebenfalls über RPC-Aufrufe als Parameter oder Rückgabewerte ausgetauscht werden. Die Verwendung von Socket-Verbindungen erlaubt allerdings eine größere Flexibilität bei der Bilddatenkommuni-

kation, so daß z.B. Streaming-Methoden besser realisiert werden können. Daher wird die Übertragung der Bilddaten über selbst aufgebaute Socket-Verbindungen dem Aufruf von Remote-Methoden über Middleware vorgezogen.

7.4.2 Umsetzung und Anwendung in einer vernetzten Rechnerumgebung

Die Hauptkomponenten des Programmcodes sind der SCA3D-Server, der SCA3D-Client und das LoI-Panel. Diese werden im Komponentendiagramm mit ihren Verbindungen untereinander dargestellt. Das Komponentendiagramm verwendet als Symbol für eine Komponente drei ineinander verschachtelte Rechtecke. Ihre Schnittstellen werden als Striche mit angehängten Kreisen gezeichnet. Die Abhängigkeiten werden als gestrichelte Linien dargestellt.

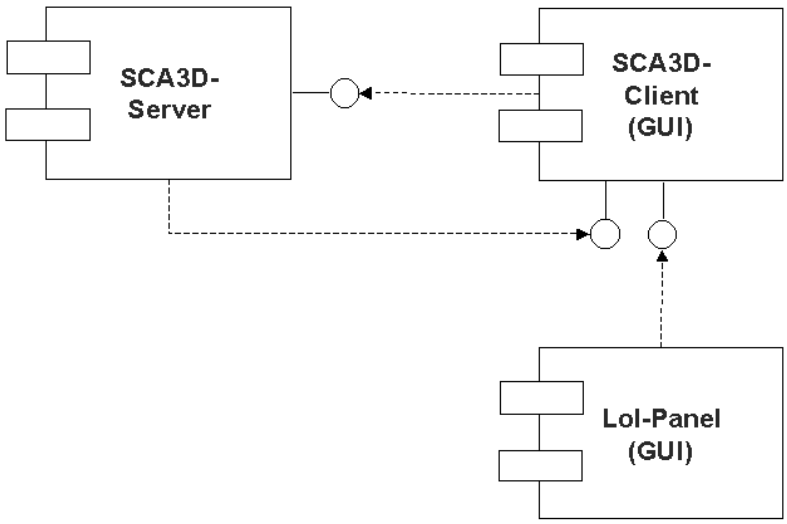

Abbildung 7.9: Komponentendiagramm der Software-Architektur SCA3D.

In Abbildung 7.9 ist das SCA3D Komponentendiagramm gezeigt, das neben dem Server, dem Client und dem LoI-Panel auch die Schnittstellen verdeutlicht. Die SCA3D-Client-Komponente greift über die CORBA-Schnittstelle auf den SCA3D-Server zu, um mit dem 3D Dokument arbeiten zu können. Umgekehrt existiert auch eine Middleware-Schnittstelle am Client, über die der Server aktiv Objektrepräsentationen in die Client-Szene einfügen kann. Die Client-Komponente enthält auch das Graphical-User-Interface, das im Entwurfskapitel als *3DVis-GUI* bezeichnet wurde. Die Schieberegler-Komponente (LoI-Panel) ist ebenfalls über ein CORBA-Interface mit dem SCA3D-Client verbunden. Das GUI mit den Schiebereglern und sonstigen Elementen ist mit Hilfe der Java-Klassenbibliothek umgesetzt.

In Abbildung 7.10 ist das Deployment-Diagramm für die SCA3D-Architektur in der oben beschriebenen Entwicklungsumgebung gezeigt. Die Server- und Client-Komponenten sind auf die zur Verfügung stehenden Rechnern verteilt. Der SCA3D-Server läuft dabei immer auf einer Unix-Maschine, während die Client-Anwendungen auf allen Plattformen zum Einsatz kommen. Es soll noch einmal betont werden, das auch der SCA3D-Server auf allen anderen Plattformen lauffähig ist. Da jedoch auch in der Praxis Server-Anwendungen häufig auf Unix-Maschinen eingesetzt werden, um deren Betriebssicherheit auszunutzen, wird hier diese Konfiguration vorgezogen.

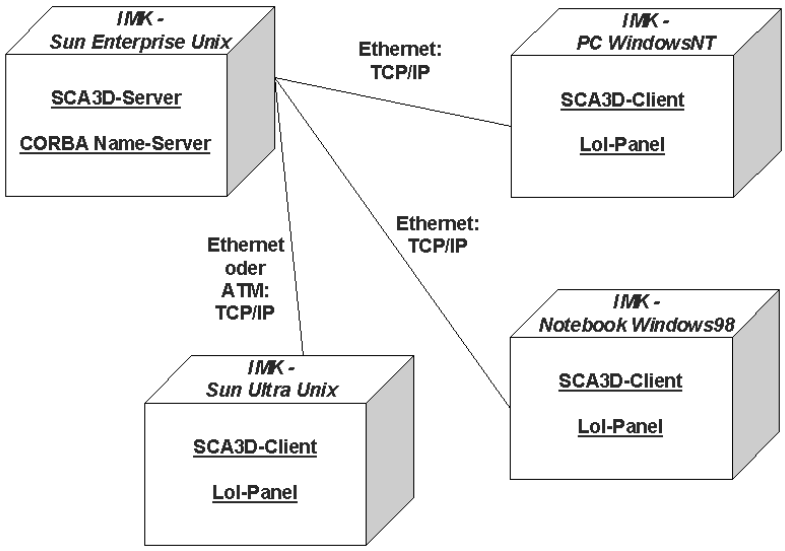

Abbildung 7.10: Deployment-Diagramm der Software-Architektur SCA3D in der Entwicklungsumgebung.

Das Deployment-Diagramm zeigt auch die weiteren Komponenten, die nicht direkt zum SCA3D-System gehören. In Abbildung 7.10 ist dies der CORBA-Name-Server, der auf dem gleichen Rechner wir der SCA3D-Server läuft. Weiterhin werden auch die Netzverbindungen zwischen den Rechnern dargestellt. Alle Rechner arbeiten im Intranet des Institutes für Medienkommunikation IMK. Das Intranet bietet zur Kommunikation Verbindungen über ein Ethernet und über eine ATM-Netz an. Alle Komponenten kommunizieren mittels des TCP/IP-Protokolls. Für Testzwecke wurden verschiedene Versionen der Software-Architektur auch außerhalb des IMK-Intranets eingesetzt. Die verwendete Technologie (WWW, CORBA, TCP/IP) erlaubt den allgemeinen Einsatz in heterogenen Netzen wie dem Internet.

7.5 Zusammenfassung

In diesem Kapitel wurde die Implementierung der SCA3D-Software-Architektur beschrieben. Nach der Vorstellung der verwendeten Software-Werkzeuge und Hardware-Komponenten wurde eine erster Prototyp, der mit MPEG-4 Videoobjekten den Zugriff auf server-seitige 3D Modelle realisierte, kurz erläutert. Dieser Prototyp setzte die Idee des Remote-Rendering zur verteilten interaktiven 3D Visualisierung ein. Dabei wurde festgestellt, daß die Verzögerungszeiten, die durch die Netzübertragung verursacht werden, für eine interaktive Anwendung im Client-Server-Modell zu groß sind. Als Weiterentwicklung führt die SCA3D-Architektur eine Kombination aus lokalem und Remote-Rendering ein. Die Implementierung der Software-Architektur wurde dann mit Hilfe eines Schichtenmodells veranschaulicht und der Einsatz in einer realen Umgebung mit Hilfe von UML-Diagrammen erläutert. Dabei wurden die Randbedingungen der Entwicklungsumgebung miteinbezogen. Zum Abschluß des Kapitels wurde die Verwendung von Threads in der Software-Architektur diskutiert und der Zusammenhang zum verwendeten Schichtenmodell aufgezeigt.

Kapitel 8

Adaptive Visualisierung in Offenen Informationsräumen

Für die in den vorhergehenden Kapiteln beschriebene Software-Architektur SCA3D soll hier ein Anwendungsszenario im Bereich Digitaler Bibliotheken entwickelt werden. Die Funktionalität der Architektur im Hinblick auf verteilte 3D Visualisierung und auf Adaptivität des Informationsgehalts auf Client-Seite soll näher untersucht werden. Dabei werden als Anwendungshintergrund offenen Informationsräume eingeführt, die technische Möglichkeiten zum Umgang mit Komplexität und Sicherheitsanforderungen von 3D Dokumenten benötigen, um ihren vollen Funktionsumfang potentiellen Benutzern zur Verfügung stellen zu können. Praktische Ergebnisse werden dargestellt, die durch Messungen in der beschriebenen Testumgebung gewonnen wurden.

8.1 Einführung: Anforderungen in offenen Informationsräumen

Verteiltes Arbeiten mit 3D Szenen als Teil von Multimedia-Dokumenten in vernetzten Umgebungen setzt die technische Unterstützung von Methoden zum Retrieval, zur interaktiver Visualisierung und zur Kommunikation voraus. Ein vernetztes Informationssystem wird hier als *Offener Informationsraum* bezeichnet, wenn der Zugriff auf Dokumente und Dienste nicht auf einen im voraus festgelegten Bereich in einem Netz und damit auf einen Kreis von Benutzer bzw. Informations-Server eingeschränkt ist.

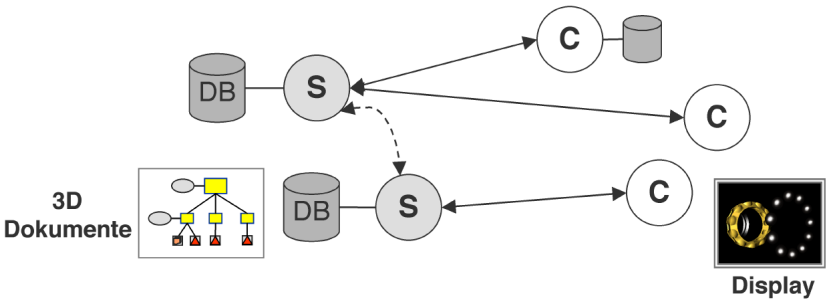

Abbildung 8.1: Visualisierung in einem offenen Informationsraum.

Weiterhin charakterisieren gleichförmige Bedingungen beim Arbeiten mit digitalen Objekten und Schnittstellen zu anderen Informationsräumen einen offenen Informationsraum, der eine Menge von Informations-Servern, Dokument-Datenbanken, Benutzern mit ihren Client-Anwendungen und Telekommunikationsnetze zwischen diesen Komponenten umfaßt. Eine solcher Informationsraum mit drei Benutzern (C), zwei Informations-Servern (S) mit Dokument-Datenbanken (DB) und einer Schnittstelle zwischen den beiden Servern ist in Abbildung 8.1 dargestellt. Zusätzlich können auch lokale Datenbanken zum Vorhalten von Information auf den Client-Rechnern vorhanden sein.

In Abbildung 8.1 ist außerdem gezeigt, wie ein Benutzer mit einem 3D Dokument arbeitet, das sich auf einem Server befindet und durch Retrieval zugreifbar gemacht wurde. Die 3D Visualisierung des Dokumentes auf Client-Seite wird hier durch Kombination von Remote-3D-Rendering und lokalem 3D-Rendering realisiert, wobei das Konzept der Skalierbaren Szenen zum Einsatz kommt. Dieses bietet den Vorteil, das die Daten des 3D Dokumentes in kontrollierbarer Form zum Benutzer übertragen werden können. Durch Verwendung eines objekt-orientierten Dokumentmodells in Form des Szenengraphen und einem Document-Request-Broker-Ansatzes wird es möglich, auf entfernte Dokumente und ihre Objekte mittels Proxy-Objekten, hier als Objektrepräsentationen bezeichnet, in verteilten Umgebungen zuzugreifen. Durch die Skalierbarkeit des Informationsgehalts der Objektrepräsentationen wird ein adaptives Verhalten bezüglich der Randbedingungen der heterogenen Umgebung realisiert. Dies wird im weiteren als Komplexitäts-Management für verteilte 3D Dokumente bezeichnet. Ein Sicherheitsmanagement für geschützte 3D Dokumente wird möglich, da Objekte eines Originaldokumentes am Server durch Objektrepräsentationen am Client vertreten werden. Die Verteilung dieser Objektrepräsentationen wird an die aktuelle Rechtesituation angepaßt.

Der untersuchte Ansatz eines Document-Request-Brokers für verteilte Dokumente ermöglicht die Realisierung der wichtigsten Forderungen an einen offenen Informationsraum, nämlich einerseits die Schaffung von gleichförmigen Bedingungen für den Zugriff auf digitale Objekte und die Bereitstellung von Schnittstellen zwischen Informationsräumen. Diese Eigenschaften werden durch die einheitliche Definition von Schnittstellen zu den Dokumenten umgesetzt. Für den Benutzer ist die Komplexität der Zugriffe auf die verteilten Dokumente transparent, d.h. im Idealfall bemerkt er keinen Unterschied zwischen lokalen und entfernten Zugriffen (Zugriffstransparenz) und der physische Ort eines Dokumentes muß dem Benutzer für einen Zugriff nicht explizit bekannt sein (Ortstransparenz). Die Ortstransparenz erlaubt die Realisierung von Schnittstellen zwischen verschiedenen Informations-Servern. Weiterhin ist es möglich, daß mehrere Benutzer sich Dokumente beim Zugriff teilen (*Concurrency-Transparenz*), ohne dieses direkt zu bemerken. Kooperation in offenen Informationsräumen wird daher vom Document-Broker-Ansatz der SCA3D Architektur als integrale Funktionalität unterstützt. Im nächsten Abschnitt wird ein Anwendungsszenario im Umfeld Digitaler Bibliotheken eingeführt, das die Eigenschaften eines offenen Informationsraumes nutzt und das mit der Software-Architektur SCA3D in Teilen umgesetzt werden soll.

8.2 Entwicklung eines Anwendungsszenarios und einer Testumgebung

Anhand eines Beispiels soll die Funktionalität, die dem Benutzer eines offenen Informationsraumes zur Verfügung steht, verdeutlicht werden. Als Beispiel dienen 3D Dokumente einer vernetzten digitalen Bibliothek, die von einer Gruppe von Studenten und einem Tutor im Rahmen eines Entwicklungsprojektes im Bereich Architektur genutzt und erzeugt werden. Eine digitale Bibliothek erlaubt es den Benutzern, bestehende Dokumente zu erkunden, neue Dokumente als Ergebnis der

Gruppenarbeit zu erzeugen und in den Dokumentbestand einzufügen. Innerhalb eines Systems einer digitalen Bibliothek sollten außerdem Möglichkeiten für verteiltes *Cut-and-Paste* mittels visueller Methoden, für die Indizierung und zur Erweiterung von Dokumenten bestehen, d.h. allgemein vertraute Funktionalität im Umgang mit Dokumenten sollte unterstützt werden.

Der Schutz von Originaldokumenten und die kontrollierte Verteilung von Kopien stellen Anforderungen dar, die bei der Entwicklung eines solchen Systems berücksichtigt werden müssen. Um den vollen Umfang der Funktionalität eines offenen Informationsraumes nutzen zu können, sollte die Arbeit mit Dokumenten sowohl in einem lokalen Netz mit eventuell spezialisierter Ausrüstung (Instituts LAN), als auch von zuhause aus mit einem Notebook-Computer über eine Modem-Verbindung möglich sein. Die Uneinheitlichkeit bezüglich der Netzanbindung bzw. der lokalen Leistungsfähigkeit der verwendeten Rechner und die Beschränkungen bezüglich bestehender Sicherheitsanforderung, wie Copyright oder Vertraulichkeitsschutz, sollten beim Entwurf einer unterliegenden Software-Architektur von Beginn an berücksichtigt und entsprechende Lösungen in das Konzept integriert werden, wie es bei der SCA3D Architektur geschehen ist.

Die Phasen während einer Arbeitssitzung, bei der ein Benutzer über eine SCA3D-Client auf seinem Rechner mit einem entfernten SCA3D-Server kommuniziert, sind im folgenden beschrieben. Bei der Ausführung treten manche Phasen eventuell mehrfach auf, wie z.B. die Evaluation der Randbedingungen, die periodisch wiederholt werden muß, da sich die Anbindung und die lokale Belastung im Laufe der Zeit ändern kann. Andere Phasen, die in variabler Reihenfolge wiederholbar sind, wie z.B. das Abspeichern von veränderten Dokumenten, werden in der folgenden Auflistung mit dem Attribut *wiederholt* gekennzeichnet.

- *Anmelden*: Ein Benutzer meldet sich zuerst beim WWW-Server der WWW-Schicht an. Die Abfrage eines Paßwortes und Paßwortschutz wird vom HTTP-Protokoll unterstützt. Wenn ein Benutzer noch keinen Zugang besitzt, wird ihm ein entsprechendes Kennwort mit Paßwort zugeordnet. Einem Benutzer ist immer ein bestimmter Benutzertyp (*User-Type*) zugeordnet. In dem hier beschriebenen Szenario gibt es Gäste, Mitglieder, Administratoren, Besitzer eines Dokumentes und Individuen, d.h. Benutzer, die einen eigenen Benutzertyp darstellen. Die Zuordnung der Rechte ist unter dem Punkt Rechteverhandlung beschrieben. Die zugehörige Information kann am Server in einer Datei oder in einer Datenbank abgelegt werden. Der SCA3D-Client-Anwendung wird die *Account*-Information zur Authentifizierung vom Benutzer mitgeteilt, entweder durch Texteingabe in ein Eingabefenster oder durch Bereitstellung einer Datei, auf die der Client zugreifen kann.

- *Retrieval*: Damit ein Benutzer mit einer Digitalen Bibliothek arbeiten kann, müssen Möglichkeiten zum Retrieval in der server-seitigen Datenbank existieren. Die Suche nach Dokumenten kann metadaten-basiert oder inhaltsbasiert erfolgen. Beim metadaten-basierten Retrieval werden zusätzliche Daten zu einem Dokument durchsucht, die in der Datenbank abgelegt wurden, um ein der Anfrage entsprechendes Dokument zugreifbar zu machen. Bei der inhalts-basierten Suche werden vorher extrahierte Merkmale der Dokumentdaten, wie z.B. Farbhistogramme bei Bildern, benutzt, um inhaltliche Übereinstimmungen zwischen Anfrage und Dokument zu finden. Während der Entwicklung der SCA3D-Architektur wurde auch eine Technik zum inhalts-basierten Suchen von 3D Modellen in relationalen Datenbanken entwickelt und umgesetzt. Die Technik und ein System ist in [Loe00b] beschrieben. Die Ausgabe des Retrievalprozesses ist eine HTML-Seite mit einer Auflistung von 3D Dokumenten, die auch eine Darstellung als gerendertes Bild enthält. Praktische Ergebnisse des Retrievals mit der vorgestellten Technik sind ebenfalls in [Loe00b] beschrieben,

- *Auswahl eines Dokumentes*: Der Benutzer kann nun durch Auswahl eines der gefundenen Dokumente, dieses Dokument in den SCA3D-Server laden und mit der interaktiven Visualisierung beginnen. Das ausgewählte Dokument wird dazu innerhalb einer HTML-Seite, die auch die nötigen Kontrollelemente für den SCA3D-Server enthält, dem Benutzer angezeigt, indem der strukturelle Aufbau wiedergegeben wird (siehe z.B. Abbildung 2.2).

- *Laden des Dokumentes in den SCA3D Server*: Durch Auslösen des *Start-Buttons*, der einem Dokument zugeordnet ist, wird der SCA3D-Server gestartet und das Dokument in den Server geladen. Falls der Server schon lief, wird das Dokument in den laufenden Server geladen. Der Benutzer startet nun lokal den SCA3D-Client, der sich mit dem Server verbindet und die verteilte Visualisierungssitzung einleitet. Zunächst ist das 3D Dokument dem Benutzer nur als Bildrepräsentation, d.h. über Remote-Rendering-Methoden, zugänglich.

- *Evaluation der Randbedingungen (periodisch)*: Sobald die Verbindung zwischen Server und Client aufgenommen ist, wird mit der Grundkonfiguration der Client-Szene die erste Performanzauswertung vorgenommen, wie sie in Abschnitt 5.8 beschrieben wurde. Dabei wird das optimale Verhältnis zwischen lokaler (LFR) und Remote-Framerate (RFR) bestimmt und entsprechend eingestellt. Nach einer festgelegten Zeitspanne wird diese Einstellung durch Messung der LFR- und RFR-Werte überprüft. Falls das optimale Verhältnis zwischen lokaler und Remote-Framerate nicht mehr erreicht wird, muß eine neue Einstellung durch eine erneute Performanzmessung ermittelt werden. Diese neue Einstellung ist dann optimal an die Randbedingungen, wie Anbindung und lokale Last, angepaßt.

- *Aushandlung und Vergleich der Rechte (wiederholt)*: Beim Laden des Dokumentes in den SCA3D-Server werden die entsprechenden IPR-Informationen in den Szenengraph als Zustand eines Eigenschaftsknotens integriert. Bei der Aktivierung eines Objektes wird die Benutzerinformation, die mit dem Remote-Document-Call (RDC) übertragen wird, am Server mit der IPR-Information der aktivierten Objektgruppe verglichen. Die entscheidenden Informationen zur Erzeugung einer Objektrepräsentation sind die sogenannte *Document-Distribution-Class* (DDC, vgl. Abschnitt 2.4.4) und die Auflösung δ, die in der IPR-Information festgelegt sind. Die Benutzerinformation enthält ebenfalls eine DDC und einen Wert für die Auflösung, die durch den Benutzertyp bestimmt werden (z.B. Gast: $DDC = ClassB, \delta = low$). Diese Informationen geben die obere Grenze der DDC und der Auflösung an, die bei der Erzeugung der Objektrepräsentation eingehalten werden muß (z.B. ein Gast kann keine ClassC-Objekte erhalten). Weiterhin gibt die Benutzerinformation den aktuellen Level-of-Information der Client-Anwendung an, der ebenfalls durch die beiden Angaben DDC und Auflösung beschrieben wird.

  Möchte ein Benutzer individuelle Rechte erwerben, um z.B. als Gast einen höhere Auflösung eines Objektes zu erhalten, so muß dies verhandelt werden. Es wäre z.B. denkbar, daß ein Dokument bei Angabe einiger Adreßangaben des Benutzers mit höherer Auflösung erhältlich ist. Zum Erwerb eines Originaldokumentes müßte eine sichere Zahlungsmethode integriert werden. Dieser Punkt soll jedoch hier nicht weiter untersucht werden. In dem vorgestellte Planungsprojekt im Architekturbereich kann vielmehr eine bestimmte Gruppe von Benutzern festgelegt werden, die automatisch Zugriff auf alle Originale der Dokumente haben, die sie selbst erzeugt und in die Digitale Bibliothek integriert haben. Genauso sind nur dieser Gruppe Schreibrechte zugewiesen, so daß sie Dokumente in die Datenbank einfügen können.

- *Interaktion mit Objekten (wiederholt)*: Der Benutzer hat verschiedene Möglichkeiten mit den

Dokumenten zu arbeiten. Die Visualisierung kann beeinflußt werden, durch Veränderung der Beleuchtung oder der Kameraparameter. Objekte können verschoben, rotiert und skaliert werden. Mit höherwertigen Methoden kann auch eine Modellierung, z.B. von parametrisierten Oberflächen, erreicht werden. Dabei wird immer mit der lokalen Objektrepräsentation als Proxy-Objekt gearbeitet. Auch neue Objekte sollen in das 3D Dokument eingefügt werden können. Dies wird durch zusätzliches Laden von Dokumenten erreicht.

- *Abspeicherung des neuen Dokumentes in der DL und einer lokalen Kopie (wiederholt)*: Nach der Bearbeitung können Dokumente in die Datenbank eingefügt werden und eventuell lokale Kopien des client-seitigen 3D Dokumentes lokal abgespeichert werden. Hat der Benutzer eine Original erworben, so wird dieses aus der Server-Datenbank gelöscht und vollständig auf dem Benutzerrechner abgelegt.

- *Schließen eines Dokumentes (wiederholt)*: Nach der Beendigung aller Bearbeitungs- und Speicherschritte kann das 3D Dokument geschlossen werden, indem der Dokumentszenengraph am Server und am Client gelöscht wird. Danach kann ein neues Dokument geladen werden.

- *Abmelden*: Wenn die Visualisierungssitzung beendet werden soll, meldet sich der Benutzer am Server ab und schließt seine Client-Anwendung. Die Benutzer-Information wird am Server verwaltet und beim nächsten Anmelden dieses Benutzers wiederverwendet.

Im den folgenden Abschnitten werden die Punkte Komplexitätsmanagement und Sicherheitsmanagement anhand dieses Anwendungsszenarios näher untersucht und praktische Ergebnisse vorgestellt.

8.3 Komplexitäts-Management für verteilte 3D Dokumente

In diesem Abschnitt wird eine Lösung zum Umgang mit Komplexität von 3D Dokumenten in dem skizzierten Anwendungsszenario unter Verwendung der SCA3D-Architektur besprochen. Dazu werden für verteilte Dokumente und für Benutzer bzw. ihre Client-Anwendungen Informationen festgelegt, die eine Klassifizierung und auf diese Weise eine sinnvolle Steuerung der Dokumentverteilung erlauben. Ein Dokument wird hier durch seinen Informationsgehalt und seine Datenmenge beschrieben. Client-Anwendungen werden Klassen zugeordnet, die ihren gemessenen lokalen und Remote-Frameraten entsprechen. Im folgenden Abschnitt wird dann diese Information mit Hilfe von Document-Distribution-Klassen für ein Sicherheitsmanagement erweitert, um den Schutz von geistigen Eigentumsrechten und der Vertraulichkeit von Daten (Copyright, vertrauliche Produktdaten, etc.) zu gewährleisten.

8.3.1 Klassifizierung von 3D Dokumenten bezüglich des Informationsgehaltes

Zur Beschreibung des Informationsgehaltes eines 3D Dokumentes wird seine Repräsentationform und die jeweilige Auflösung verwendet. Der Einfachheit halber werden die Objektrepräsentationen, die von SCA3D unterstützt werden, durch folgenden Abkürzungen gekennzeichnet: $R1$: parametrische Modelle, $R2$: polygonale Netze, $R3$: Bounding-Box, $R4$: Bilddaten. Der Informationsgehalt wird dadurch bestimmt, wieviel Information über das 3D Dokument der Benutzer durch die Repräsentationform gewinnen kann. Parametrische Modelle besitzen daher hier den größten Informationsgehalt und Bilddaten den geringsten.

Die Auflösung wird entsprechend der jeweiligen Repräsentationform angegeben. Also z.B. mittlere Maschengröße bei 3D-Netzen oder bei Bildern Bildpunkte pro Längeneinheit (dpi). Bei parametrisierten Modellen wird davon ausgegangen, daß die Parameter vollständig übertragen werden. Wenn der Client also das Basiswerkzeug zur Erzeugung eines solchen Objektes kennt, hat er mit der Angabe der Parameter die Möglichkeit, das Objekt beliebig genau darzustellen und zu reproduzieren. Für ein parametrisch dargestelltes 3D Objekt wird hier daher nur ein möglicher Wert für den Informationsgehalt angenommen. Im Fall von 3D-Netzen hat ein Objekt mit einer höheren Auflösung einen höheren Informationsgehalt als ein Objekt mit geringerer Auflösung. Dabei werden keine numerischen Werte für den Informationsgehalt verwendet, sondern die beiden getrennten Angaben der Kennung ($R1$-$R4$) und der Auflösung (δ_R) zum Vergleich benutzt.

Bei der Verteilung von 3D Dokumenten ist nicht nur der Informationsgehalt wichtig, sondern besonders auch die damit verbundene Datenmenge, die übertragen werden muß. Ein 3D Objekt, das von einer Visualisierungsanwendung dargestellt wird, kann auf dem Benutzerdisplay mit der gleichen Auflösung erscheinen, also die gleiche Rechenkapazität in Anspruch nehmen, obwohl die vorher übertragene Datenmenge sehr unterschiedlich ist. Da auch parametrische Objekte bei der Visualisierung in 3D-Netze mit einer bestimmten Auflösung umgewandelt werden, ist die Repräsentationform in diesem Fall hauptsächlich entscheidend für die Übertragung. Parametrische Modelle sind aufgrund ihrer meist geringeren Datenmenge, die übertragen werden muß, oft von Vorteil, wenn eine schwache Netzanbindung vorliegt. Allerdings liegen viele 3D Modelle von vornherein als 3D-Netz vor, lassen sich nicht einfach in parametrisierter Form darstellen oder sollen eben nicht mit vollen Informationsgehalt übertragen werden, sodaß dieser Vorteil nicht immer nutzbar ist.

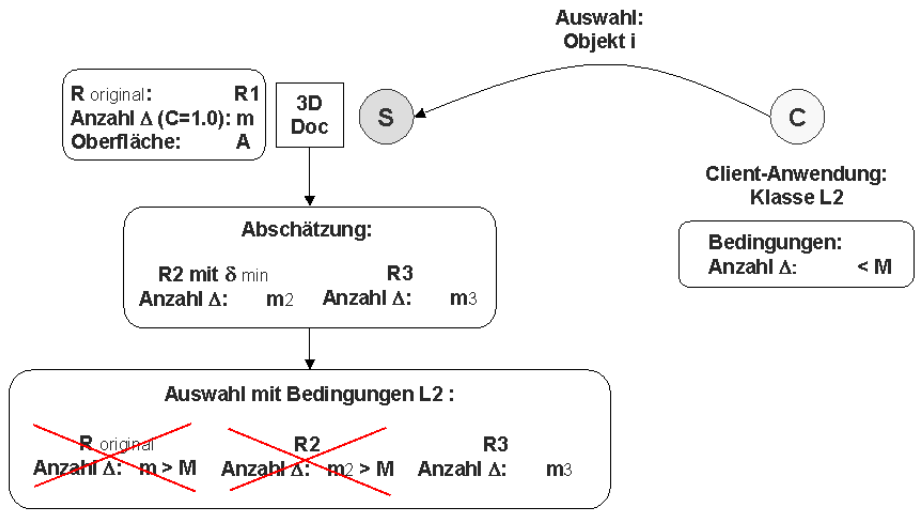

Abbildung 8.2: Auswahl einer geeigneten Repräsentationsform.

Die Datenmenge eines 3D Dokumentes ist im Falle von 3D-Netzen proportional zur Anzahl der Dreiecke, mit denen die Oberfläche modelliert ist. Aus der Anzahl der Polygone multipliziert mit dem Speicherbedarf pro Polygon kann die Datenmenge ungefähr berechnet werden, wobei zusätzliche Information, wie Materialparameter oder Metadaten, nicht erfaßt werden. Die Anzahl der Polygone hängt von der Auflösung und von der Größe der Oberfläche ab. Die Auflösung ist in Anzahl von Dreiecken pro Flächeneinheit gegeben. Wenn ein Benutzer ein 3D Objekt selektiert,

werden die wichtigsten Metadaten, wie Repräsentationsform R_i, benötigte Anzahl der Dreiecke bei der Darstellung, Datenmenge und Oberfläche ermittelt. Dann wird eine Abschätzung der gleichen Metadaten für dieses Objekt in einer anderen Repräsentationsform vorgenommen. Durch den Vergleich der Bedingungen der Leistungsklasse (*L-Klasse*), die der Client-Anwendung bei der Performanzmessung zugeordnet wurde, wird die passende Objektrepräsentation ausgewählt.

In Abbildung 8.2 ist der Prozeß der Auswahl einer geeigneten Repräsentationsform eines 3D Dokumentes für eine bestimmte L-Klasse gezeigt. Die Repräsentationsform, die Anzahl der Polygone und die Gesamtoberfläche des Originaldokumentes wird bestimmt, die beiden letzteren Werte in der Regel näherungsweise. Dann wird ein Abschätzung der gleichen Daten für die in Frage kommenden Repräsentationformen vorgenommen. Im Fall von 3D-Netzen wird eine minimale Auflösung δ_{min} gefordert, um ein Übergangskriterium zur nächst niedrigeren Repräsentationform festzulegen. Es wird angenommen, daß unterhalb von δ_{min}, der Informationsgehalt gegenüber einer Bounding-Box-Repräsentation nicht wesentlich besser ist. Die obere Grenze der Auflösung δ_{max} wird durch die maximale Anzahl von Dreiecken bestimmt, die durch die Leistungsklasse der Benutzeranwendung festgelegt wird. Mit Hilfe dieser Grenzen wird die passende Objektrepräsentation für das selektierte Objekt und die aufrufende Client-Anwendung ermittelt. In gezeigten Fall wird die Bounding-Box-Information des Objektes übertragen ($R3$), da eine 3D-Netz mit der geforderten Auflösung eine zu große Menge an Polygonen enthalten würde und daher der Client zu stark belastet würde. Im nächsten Abschnitt werden die sogenannten L-Klassen näher vorgestellt.

8.3.2 Einordnung von Client-Anwendungen in Leistungsklassen

Die Client-Anwendungen werden entsprechend der beim Performanztest gemessenen Werte von lokaler und Remote-Framerate in vier Klassen (L-Klassen) eingeteilt, die vier Quadranten im LFR-RFR-Diagramm entsprechen (siehe Abbildung 8.3). Jeder L-Klasse werden dann Objektrepräsentationen mit bestimmten Informationsgehalten zugeordnet, wobei eine vorgenommene Zuordnung sich im Verlauf des Arbeitsprozesses als sinnvoll herausstellen muß.

Automatisch wird nun ein Bereich des Schiebereglers markiert, innerhalb dessen die Bedingungen der Leistungsklasse erfüllt werden. Der Benutzer kann dann durch Einstellung innerhalb dieses Bereiches eine Reglerposition wählen und das selektierte Objekt anfordern, das dann übertragen und in die Client-Szene eingefügt wird. Ein Benutzer kann aber auch von diesen Empfehlungen abweichen und eine anderen Repräsentationsform wählen, solange er nicht die Sicherheitsbedingungen verletzt, die später beschrieben werden.

In Abbildung 8.3 ist eine mögliche Konfiguration der L-Klassen im LFR-RFR-Diagramm für eine Anwendungsszenario dargestellt. Diese Konfiguration ist durch einen maximalen Wert für die Anzahl der Polygone pro selektiertes Objekt und die geforderten Frameraten der Benutzeranwendung angegeben. Die Grenzen LFR_0 Frames/s und RFR_0 Frames/s legen die vier in Abbildung 8.3 verwendeten L-Klassen fest. Die Grenzen sind dabei als Erfahrungswerte aus den Messungen in der Testumgebung entstanden. Client-Anwendungen werden nach einem Performanztest, in eine der vier Klassen eingeordnet. Die Bedingungen der Klasse werden dann für die Zuteilung von Objektrepräsentationen benutzt.

8.3.3 Realisierung mittels der OpenInventor-Klasse *SoComplexity*

Im folgenden soll eine Realisierung für einen eingeschränkten Anwendungsfall beschrieben und innerhalb der SCA3D-Architektur getestet werden. Die Einschränkung besteht darin, daß aus Originaldokumenten nur geometrische Basisobjekte (Sphere-, Cube-, Cone-Objekt, NURBS-Flächen,

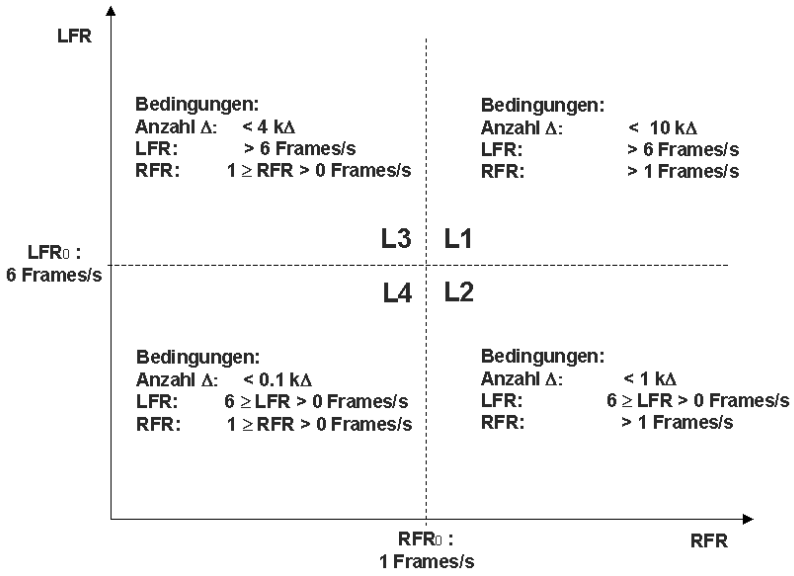

Abbildung 8.3: Einteilung des LFR-RFR-Diagramms in Leistungsklassen.

etc.), die sich im OpenInventor-Format parametrisch darstellen lassen, mit skalierbarem Informationsgehalt aktiviert werden können. 3D-Netze innerhalb des Originaldokumentes werden entweder wieder als 3D-Netze mit der gleichen Auflösung aktiviert oder können als Bounding-Box-Repräsentation in die Client-Szene eingefügt werden. Weiterhin werden hier nur Übergänge von parametrischen Repräsentationsformen ($R1$) zu 3D-Netzen ($R2$) oder zu Bounding-Box-Repräsentation ($R3$) näher untersucht. Das Originalobjekt liegt also als $R1$-Objekt vor und wird bei der Aktivierung entweder in ein Dreiecksnetz mit geeigneter Auflösung oder alternativ in eine $R3$-Repräsentation umgewandelt. Mit diesen Einschränkungen lassen sich trotzdem die wichtigsten Eigenschaften und Methoden des Komplexitätsmanagements der SCA3D-Architektur untersuchen, wobei gleichzeitig die entscheidenden Konzepte klarer herauskommen.

Eine Möglichkeit das beschriebene Komplexitätsmanagement mit OpenInventor umzusetzen, ist die Verwendung der Klasse *SoComplexity*. Diese Klasse, deren Instanzen als Knoten in den Szenengraph eingebunden werden, erlaubt die Steuerung der Auflösung des Dreiecksnetzes, das bei der Darstellung von parametrischen Objekten verwendet wird. Z.B. werden auch Basisobjekte wie eine Kugel oder ein Konus, die im OpenInventor-Format durch die Parameter Radius bzw. Grundradius und Höhe angegeben werden, vor der Darstellung auf dem Bildschirm in 3D-Netze umgewandelt und weiterverarbeitet. Die Auflösung ist normalerweise mit einem Komplexitätswert von $C = 1.0$ verknüpft. Indem nun ein Complexity-Knoten vor das betreffende Objekt im Szenengraphen plaziert und dem Komplexitätswert eine Zahl zwischen 0.0 und 1.0 zugewiesen wird, kann die Auflösung dieses 3D-Netzes schrittweise reguliert werden.

Der Zusammenhang zwischen Komplexitätswert und Anzahl der Dreiecke ist für verschiedene Objekte unterschiedlich. Eine Kugel wird z.B. bei einem maximalen Komplexitätswert mit sehr viel mehr Dreiecken dargestellt als ein Konus. Dieser Zusammenhang ist für OpenInventor Kugel- und Konus-Objekte gemessen worden und in Abbildung 8.4 dargestellt. Es ist anzumerken, daß die Anzahl der Dreiecke pro Objekt nicht von der Größe der Objekte abhängt, d.h. bei einem fe-

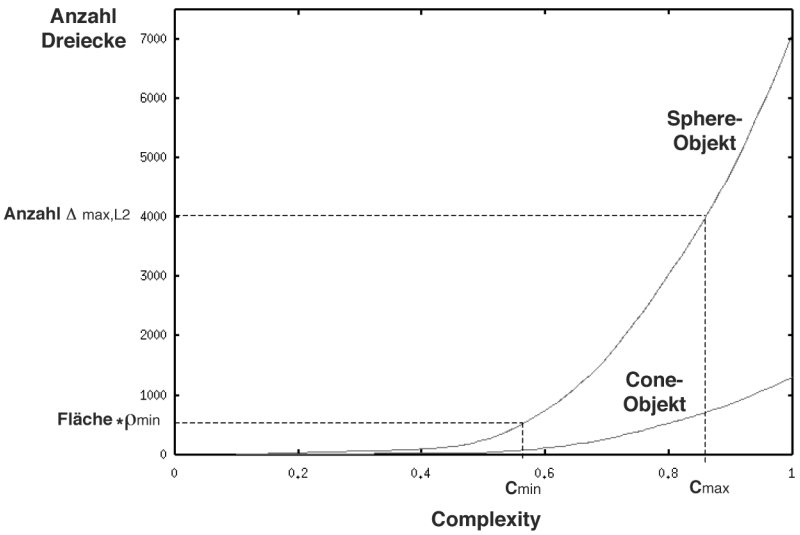

Abbildung 8.4: Zusammenhang zwischen Complexity-Wert und Dreiecksanzahl für verschiedene OpenInventor-Objekte.

sten Komplexitätswert wird ein Kugelobjekt mit dem Radius $r = 1.0$ mit der gleichen Anzahl von Dreiecken dargestellt, wie eine Kugel mit dem Radius $r = 10.0$. Daraus folgt, daß mit der SoComplexity-Klasse nicht direkt die Auflösung beliebiger Objekte kontrolliert werden kann, sondern es muß jeweils eine Umrechnung zwischen Komplexitätswert und Auflösung mit Hilfe der Angaben über Objekttyp und Oberfläche vorgenommen werden. Die Ermittlung von Komplexitätswerten, die zu einer Auflösung eines Objektes gehören, ist graphisch ebenfalls in Abbildung 8.4 gezeigt. Um den Komplexitätswert zu ermitteln, wird die geforderte Auflösung ρ mit dem Oberflächenwert multipliziert und damit die Gesamtanzahl der Dreiecke des 3D-Netzes berechnet. Der zugehörige Komplexitätswert $C_{Fläche\star\rho}$ kann dann anhand der Kurve abgelesen werden. Die maximale Anzahl von Dreiecken pro Objekt, die einer L-Klasse entspricht, kann direkt dem Komplexitätswert C_{max} zugeordnet werden. In der Software-Architektur SCA3D wird die Zuordnung der Komplexitätswerte zu der Dreiecksanzahl für verschiedene Objekte als Parameterliste bekannt gemacht.

In Abbildung 8.6 sind Objektrepräsentationen einer parametrischen Oberfläche gezeigt, die bei der Aktivierung in den Interaktionsraum des Benutzers eingefügt wurden. Es handelt sich dabei um Dreiecksnetze mit steigendem Komplexitätswert zwischen $c = 0.1$ und $c = 0.9$. Außerdem ist noch das Originalobjekt als NURBS-Fläche im Interaktionsraum gezeigt. Um für ein 3D Objekt, das von einem Benutzer aktiviert wird, eine Objektrepräsentation als 3D-Netz mit einer bestimmten Auflösung zu erzeugen, wird in die Gruppe des selektierten Objektes, die als Kopie im Interface-Objekt vorliegt, ein SoComplexity-Knoten eingefügt und der Komplexitätswert so eingestellt, das bei der Erzeugung des 3D-Netzes die passende Auflösung erzielt wird. Das 3D-Netz selbst wird durch Abtasten des Originalobjektes beim Aufruf eines OpenInventor-*SoTriangleCB* erzeugt, wobei die Eckpunktkoordinaten der Abtastpunkte gespeichert werden. Diese Eckpunkte werden dann z.B. mit Hilfe von *SoTriangleStripSet* in ein zusammenhängendes Dreiecksnetz eingefügt. Die Methoden des Interface-Objektes, die zum Aufruf des *SoTriangleCB* benötigt werden, sind in Abbildung 8.5 als Java-Quelltext gezeigt.

```
public static void scanObject(SoNode root)
{
    SoCallbackAction myAction = new SoCallbackAction();
    SoTriangleCB tricb = new SoTriangleCB ("InterfaceObject", "printTriangleCallback");
    myAction.addTriangleCallback(SoNode.getClassTypeId(), tricb, null);
    myAction.apply(root);
}

public static void printTriangleCallback(Object userData, SoCallbackAction action,
                                         SoPrimitiveVertex vertex1,
                                         SoPrimitiveVertex vertex2,
                                         SoPrimitiveVertex vertex3)
{
    printVertex(vertex1);
    printVertex(vertex2);
    printVertex(vertex3);
}

public static void printVertex( SoPrimitiveVertex vertex)
{
    SbVec3f point = vertex.getPoint();

    if (index <ver){
        vertexPositions[index][0] = point.opindex(0);
        vertexPositions[index][1] = point.opindex(1);
        vertexPositions[index][2] = point.opindex(2);
        index++;
    }else {
        ;
    }

}
```

Abbildung 8.5: Quelltext der Methoden zum Abtasten eines 3D Objektes.

Um Kriterien für die Übergänge von einer Objektrepräsentation in eine andere zu finden, werden neben den Bedingungen aus den L-Klassen der Client-Anwendungen noch weitere Angaben über die einzelnen Level-of-Information benötigt. Das Ziel ist die Zuordnung von Objektrepräsentation mit verschiedenen Auflösungen auf die gesamte LoI-Skala, wie in Abbildung 8.7 dargestellt. Um diese Zuordnung für die Objektrepräsentationen $R1$ bis $R3$ vorzunehmen, werden die oben beschriebenen Komplexitätswerte verwendet. Am oberen Ende der LoI-Skala befinden sich die parametrischen Objektrepräsentationen, die bei der Visualisierung mit einem Komplexitätswert $C = 1.0$ verarbeitet werden. Dann folgen die 3D-Netzrepräsentationen, die Komplexitätswerten zwischen $C = 1.0$ und $C = 0.1$ zugeordnet werden. Sowohl parametrische, als auch 3D-Netzrepräsentationen können also den Komplexitätswert $C = 1.0$ besitzen. Die Bounding-Box-Repräsentationen ($R3$), die als nächstes folgen, werden mit dem Komplexitätswert $C = 0.0$ verknüpft.

Im Falle von 3D-Netzen wird eine Mindestauflösung von $\rho_{min} = N\frac{\Delta}{Flächeneinheit}$ festgelegt, um ein Kriterien für die Übergänge von $R2$ nach $R3$ zu erhalten. Aus dieser Bedingung und der maximalen Anzahl von Dreiecken pro Objekt der L-Klassen ergeben sich minimale und maximale Komplexitätswerte C_{min} und C_{max}, die den Bereich der optimalen LoI-Positionen für eine Client-Anwendung markieren. Für $R1$-Repräsentationen gilt $C_{min} = C_{max} = 1.0$, d.h. wenn eine Client-Anwendung die maximale Auflösung, mit der eine parametrische Objekt angezeigt wird, nicht mit den Bedingungen der L-Klasse vereinbaren kann, wird ein Übergang von $R1$ nach $R2$ vorgenommen.

Wenn ein Benutzer nun mit einem entfernten 3D Dokument einer digitalen Bibliothek arbeitet, werden bei der Selektion eines Objektes zwei Bereiche auf der Skala des LoI-Reglers makiert (siehe Abbildung 8.7). Einerseits wird der optimale Bereich für die Komplexität von Objektrepräsentationen in grüner Farbe gekennzeichnet und die LoI-Position automatisch in diesem Bereich eingestellt. Zum anderen wird eine Bereich rot markiert, der anzeigt, welche Objektrepräsentatio-

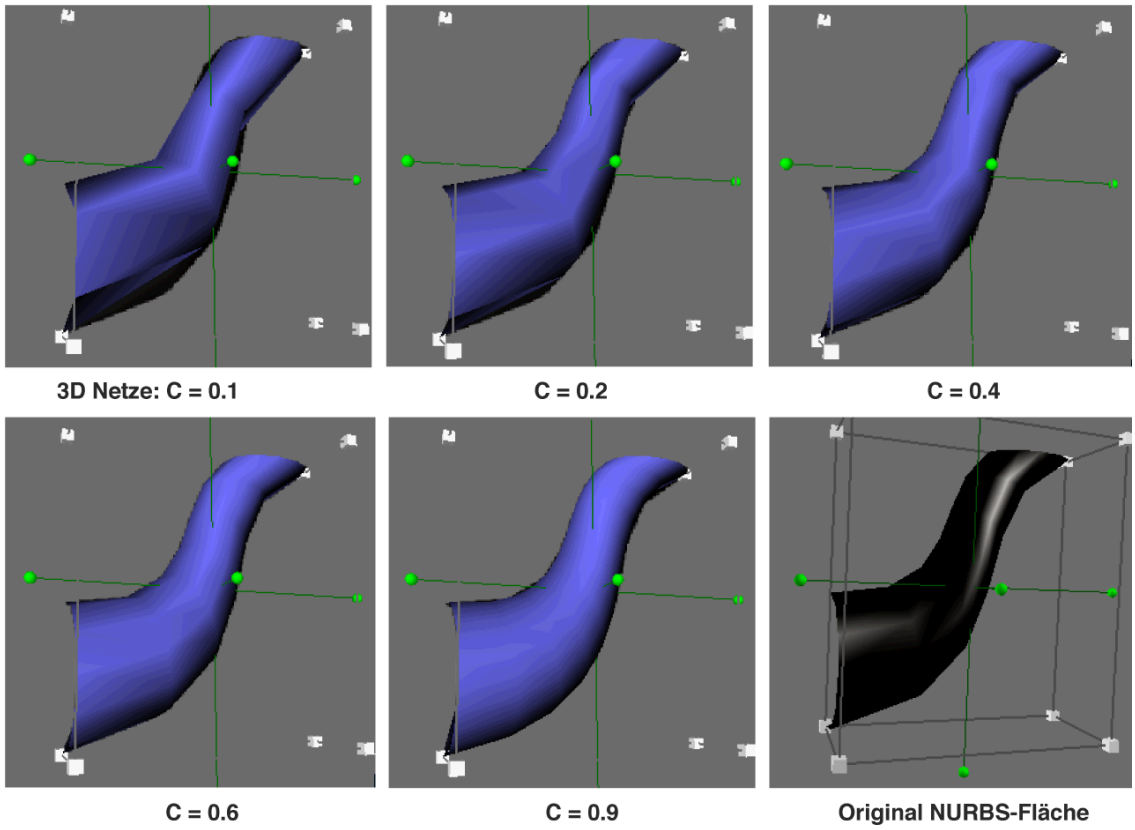

Abbildung 8.6: Repräsentationen einer NURBS-Fläche als 3D Netz mit steigeder Komplexität und als Original. Quelle des Modells: *Teapot* im Original von M.Newell.

nen aus Gründen fehlender Rechte nicht übertragen werden dürfen. Die Positionen dieses Bereiches sind für den Benutzer gesperrt.

8.3.4 Praktische Ergebnisse zum adaptiven Verhalten von SCA3D

In diesem Abschnitt sollen Ergebnisse im Bezug auf das adaptive Verhalten von SCA3D, die in der Testumgebung gewonnen wurden, vorgestellt und diskutiert werden. Zuerst wird die Leistungsbewertung und das dynamische Verhalten von Client-Anwendungen bezüglich der lokalen und Remote-Frameraten näher untersucht. Die Zuordnung von L-Klassen zu Client-Rechnern in der Testumgebung, die Anpassung der Frameraten und die wiederholte Auswertung durch Performanztests wird hier beschrieben. Danach folgt ein Unterabschnitt, in dem das Komplexitätsmanagement anhand von Messungen während des Arbeitsprozesses erläutert wird. Weitere Ergebnisse zum adaptiven Verhalten des SCA3D-Architektur sind auch in [LFe01] vorgestellt.

Dynamisches Verhalten von Client-Anwendungen im LFR-RFR-Diagramm

Anhand von Messungen mit drei Testrechnern soll eine Zuordnung von L-Klassen vorgenommen und die Bewertung anhand der Performanzmetrik aus Kapitel 5 dargestellt werden. Damit bei sich

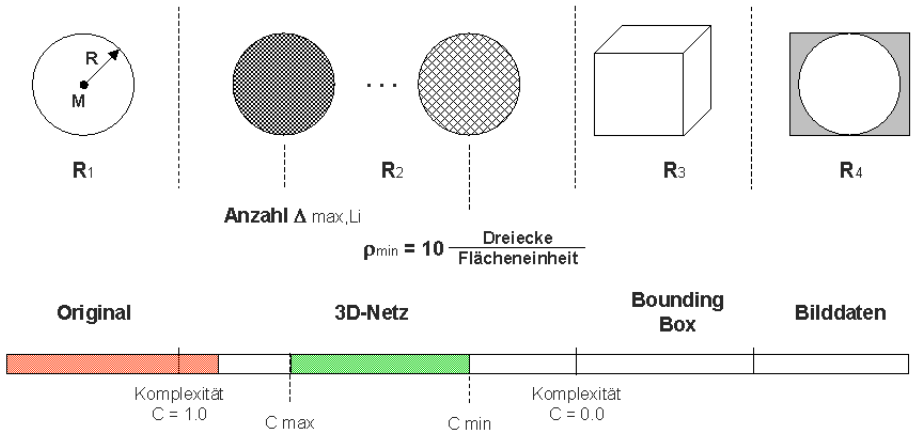

Abbildung 8.7: Verwendung des Complexity-Wertes zur Einteilung der LoI-Skala ($\rho_{min} = 10$ Dreiecke pro Flächeneinheit).

ändernden Bedingungen im Hinblick auf die Anbindungsbandbreite und die lokale Belastung eine Anpassung der verteilten 3D Visualisierung erfolgen kann, müssen die Raten, mit denen am Server Bilder gesendet werden und am Client als Textur abgebildet werden, während des Arbeitens angepaßt werden. Dies geschieht mit Hilfe der im Performanztest bestimmten LFR-RFR-Kurve, die charakteristisch für einen Client-Rechner unter den vorliegenden Bedingungen ist. Wenn sich die Bedingungen über ein festgelegtes Maß hinaus verändern, wird eine neue Auswertung der aktuellen Performanz vorgenommen, um eine neue Charakteristik zu erhalten. Diese Schritte werden im Folgenden näher untersucht:

- *Zuordnung von L-Klassen*: Die Testrechner sind ein Windows-Notebook, ein Windows-PC und eine Unix-Workstation, mit den Eigenschaften wie sie in Abbildung 7.3 beschrieben wurden. Für alle drei Rechner wurde mit der Grundkonfiguration der Client-Szene eine Performanzauswertung vorgenommen und ein optimales Verhältnis von lokaler und Remote-Framerate bestimmt. Diese optimalen Punkte sind in Abbildung 8.8 durch die Vektoren vom Ursprung des LFR-RFR-Diagramms zum Koordinatenpunkt $(LFR, RFR)_{opt}$ dargestellt. Zur Einordnung von Client-Rechnern in L-Klassen wurden die Werte für LFR_0 ($5 Frames/s$) und RFR_0 ($1 Frames/s$) festgelegt. Diese Werte wurden hier so gewählt, daß in den Klassen $L1$ (Windows-Notebook), $L2$ (Unix-Workstation) und $L4$ (Windows-PC) jeweils ein Repräsentant vorhanden war. Die ermittelten Werte für lokale und Remote-Frameraten der einzelnen Client-Rechner sind in Abbildung 8.12 aufgelistet.

- *Anpassung von Server-Framerate und Texture-Mapping-Rate*

  Durch Veränderung der verfügbaren Bandbreite bzw. der lokalen Last, hervorgerufen durch eine Erweiterung der Client-Szene oder durch andere Prozesse auf dem Rechner, werden sich die Werte der lokalen und Remote-Frameraten während des Arbeitsprozesses ebenfalls ändern. Damit das verteilte System dynamisch auf diese Änderungen reagieren kann, müssen die Bilderzeugungsraten am Server (SFR) und die lokale Texture-Mapping-Rate (TMR) an die neuen Bedingungen angepaßt werden. Dafür werden in der SCA3D-Architektur, nach-

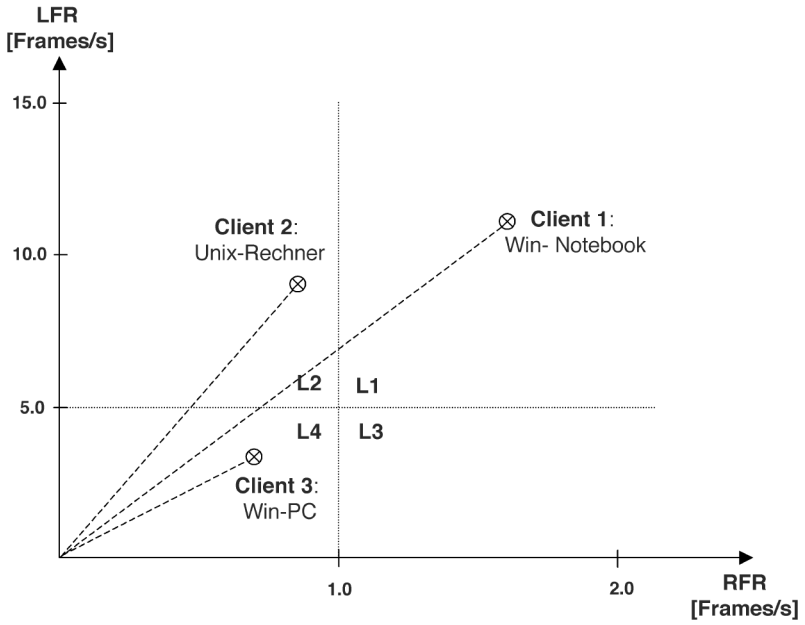

Abbildung 8.8: Klassifizierung der Testrechner.

dem eine Performanztest ausgeführt wurde, die lokale und Remote-Framerate in regelmäßigen Abständen gemessen und anhand der LFR-RFR-Kurve eine Anpassung von TMR und SFR vorgenommen. Dabei wird aus dem LFR-RFR-Diagramm, das durch schrittweise Erhöhung der Texture-Mapping-Rate während der Messung entstanden ist, der entsprechende Wert für die Texture-Mapping-Rate als Parameter bestimmt. Die entsprechenden Meßwerte und der TMR-Parameter werden dafür intern gespeichert und sind für die letzte Performanzmessung zugreifbar. Die Server-Framerate SFR wird dann über das CORBA-Interface entsprechend dem Wert von TMR automatisch angepaßt, wobei die beiden Werte hier gleichgesetzt werden. Wenn mehr als ein Client versorgt werden muß, wird die Server-Framerate an die maximale Texture-Mapping-Rate angepaßt und die Füllraten der im Server integrierten Bild-Buffer für die anderen Client-Anwendungen entsprechend reguliert.

- *Wiederholte Auswertung der Performanz*

  Wenn die Bandbreite und lokale Last zu stark von den Bedingungen abweicht, die während des Performanztests vorgeherrscht haben, dann wird eine erneute Auswertung der Performanz veranlaßt. Um ein Kriterium für die Wiederholung zu finden, wird wiederum das LFR-RFR-Diagramm herangezogen, wie in Abbildung 8.9 für einen Testrechner dargestellt. Dafür wird die Steigung der LFR-RFR-Kurve so ausgewertet, daß ein Bereich (grüner Bereich in Abbildung 8.9 (rechts)) bestimmt wird, in dem die Steigung an den ausgewerteten Punkten noch nicht unter 30% des maximalen Wertes gesunken ist. Dieser Bereich wird auf Bereiche für LFR und RFR abgebildet, die in Abbildung 8.9 (links) ebenfalls grün markiert sind. Innerhalb dieser Bereiche behält die Performanzmessung ihre Gültigkeit, während eine erneute Auswertung nötig wird, wenn die Werte der Frameraten außerhalb dieser Bereiche liegen. Dies ist in Abbildung 8.9 (links) ebenfalls angemerkt. Die dargestellten Kurven sind

durch Interpolation der Meßwerte mit dem Programm *Gnuplot* entstanden.

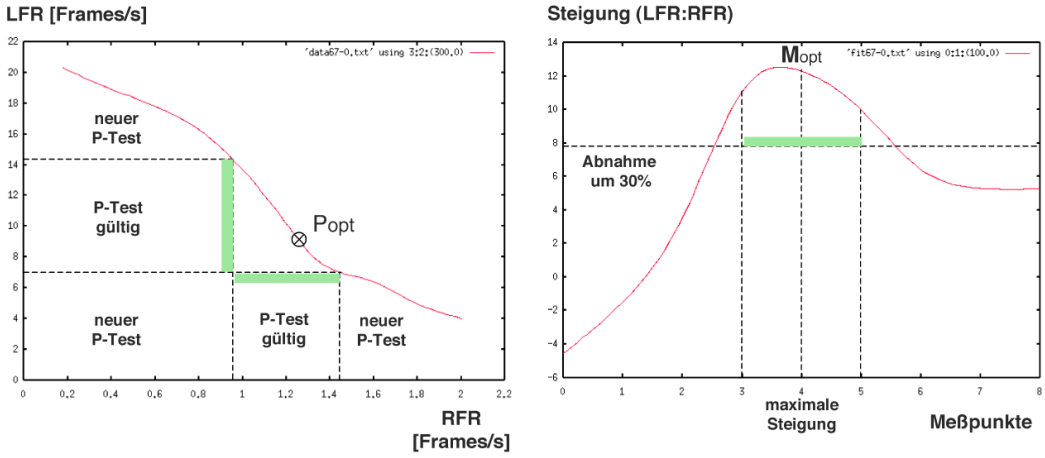

Abbildung 8.9: Bestimmung eines stabilen Bereiches für die Leistungsbewertung von Client 1 (Win-Notebook).

Die beschriebene Auswertung der Performanz ist für die anderen beiden Rechner der Test/-umgebung in den Abbildungen 8.10 und 8.11 dargestellt. Es ist zu erkennen, daß die optimalen Einstellungen für die Frameraten zwischen den Rechnern variieren. Der typische Verlauf der LFR-RFR-Kurven mit einer maximalen Steigung und einer Abflachung in den äußeren Bereichen ist jedoch bei allen Testrechnern zu beobachten. Dieser Verlauf kann durch die Leistungsgrenze des Servers bei der Bilderzeugung bzw. -übertragung und die maximale lokale Framerate ohne Texture-Mapping erklärt werden. Im Bereich relativ hoher Remote-Frameraten wird auch bei weiterer Steigerung von TMR und SFR keine Erhöhung der Rate erzielt, mit der Texturen übertragen und angezeigt werden. Daher kann der Client-Rechner eine konstante Arbeitsmenge für das lokale 3D-Rendering verwenden, wodurch sich die Abflachung am rechten Rand der LFR-RFR-Kurve ergibt. Für sehr geringe Bildraten vom Server, muß die Client-Anwendung eine verschwindend geringe Arbeitsleistung für das Texture-Mapping aufwenden, sodaß hier die maximale lokale Framerate für eine 3D Szene ohne Texture-Mapping zu einer Abflachung der Kurve am linken Rand führt. Zwischen diesen Rändern liegt der Bereich, innerhalb dessen lokale und Remote-Visualisierung optimal kombiniert werden können, da die jeweilige Last am besten verteilt wird.

Einteilung der Level-of-Information-Skala: C_{min}, C_{max}

Die oben beschriebene Klassifizierung der Testrechner führt zu einer dynamischen Einteilung der LoI-Skala des Schiebereglers auf Client-Seite. Bei der Selektion eines Objektes des server-seitigen 3D Dokumentes wird die Anzahl der benötigten Dreiecke für dieses Objekt mit verschiedener Komplexität abgeschätzt, indem die Oberfläche ermittelt und mit der Mindestauflösung multipliziert wird. Für einfache Objekte wie eine Kugel oder einen Konus läßt sich die Oberfläche genau berechnen, für kompliziertere Oberflächen wird eine Abschätzung durch die Oberfläche eines Bounding-Volumen, z.B. die Bounding-Box, herangezogen. Aus der Oberfläche und der Mindestauflösung kann dann der minimale Komplexitätswert c_{min} ermittelt werden, wie es in Abschnitt 8.3.3 be-

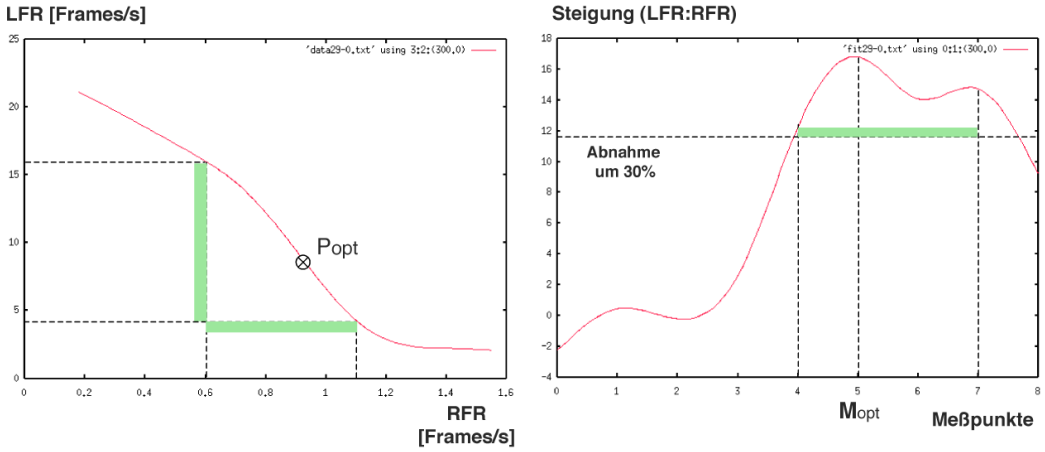

Abbildung 8.10: Leistungsbewertung für Client 2 (Unix-Rechner).

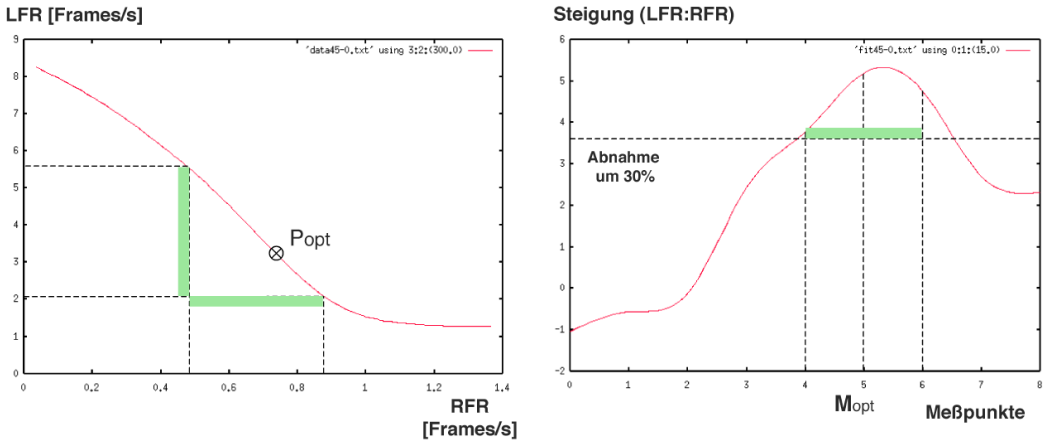

Abbildung 8.11: Leistungsbewertung für Client 3 (Win-PC).

schrieben wurde. Der Zusammenhang von Komplexitätswert und Anzahl von Dreiecken muß dafür mit Hilfe des OpenInventor API im voraus bestimmt werden. Der maximale Komplexitätswert c_{max} ergibt sich aus der oberen Grenze für die Datenmenge pro Objekt der jeweiligen L-Klasse. In Abbildung 8.12 sind die Werte für c_{min} und c_{max} für die verschiedenen Rechner bei der Aktivierung einfacher Objekte mit unterschiedlichen Abmessungen aufgelistet.

	LFR	RFR	L-Klasse	Sphere (R=2.0)	Sphere (R=10.0)	Cone (R=0.3, H=2.0)
Client 1	11,30	1,20	L1	cmin = 0.6	cmin = 1.0	cmin = 0.4
				cmax = 1.0	cmax = 1.0	cmax = 1.0
				Übergang: R1->R1;R2	Übergang: R1->R1;R2	Übergang: R1->R1;R2
Client 2	9,00	0,90	L2	cmin = 0.6	cmin = 1.0	cmin = 0.4
				cmax = 0.8	cmax = 0.8	cmax = 1.0
				Übergang: R1->R2	Übergang: R1->R3	Übergang: R1->R1;R2
Client 3	3,25	0,75	L3	cmin = 0.6	cmin = 1.0	cmin = 0.4
				cmax = 0.3	cmax = 0.3	cmax = 0.5
				Übergang: R1->R3	Übergang: R1->R3	Übergang: R1->R2

Abbildung 8.12: Bestimmung der Schranken c_{min} und c_{max} und der entsprechenden LoI-Übergänge.

Aus den Werten von c_{min} und c_{max} ergeben sich die Übergänge zwischen verschiedenen Objektrepräsentationen während des Arbeitens mit einem 3D Dokument für die verschiedenen Benutzer. Die Übergänge sind ebenfalls in Abbildung 8.12 genannt. Die Bezeichnung $R_i \rightarrow R_j$ meint dabei einen Übergang von der Objektrepräsentation R_i (z.B. der parametrischen Darstellung) zur Objektrepräsentation R_j (z.B. einer Darstellung als 3D Netz).

Die Übergänge werden dabei nach bestimmten Regeln veranlaßt. Es gilt, daß für $c_{min} = c_{max} = 1.0$ bzw. $c_{min} < c_{max} = 1.0$ Übergänge $R1 \rightarrow R1$ bzw. $R1 \rightarrow R2$ möglich sind. Wenn gilt, $c_{min} \leq c_{max} < 1.0$ ist nur der Übergang $R1 \rightarrow R2$ möglich. Schließlich ist für Komplexitätswerte, die der Bedingung $c_{min} > c_{max}$ folgen, nur der Übergang $R1 \rightarrow R3$ erlaubt. Dies folgt daraus, daß c_{min} nicht den Wert Null annehmen kann, da die vorgegebene Mindestauflösung dies verbietet. In Abbildung 8.12 ist zu erkennen, daß die Übergangskriterien dynamisch sind, d.h. von der L-Klasse einer Client-Anwendung und von dem Typ bzw. den Ausmaßen des selektierten Objektes abhängen.

Wenn ein Benutzer eine Objekt aktiviert, wird eine Objektrepräsentation mit einer günstigen Auflösung in der passenden Darstellungsform in seine Client-Szene integriert. Die Position des Schiebereglers kann dabei zwischen den Werten c_{min} und c_{max} gewählt werden. Die automatische Positionierung erfolgt dabei so, daß der mittlere Komplexitätswert zwischen c_{min} und c_{max} gewählt wird.

8.4 Sicherheits-Management durch variablen Informationsgehalt von Objektrepräsentationen

Nachdem im vorigen Abschnitt eine Umsetzung für das Komplexitätsmanagement innerhalb der SCA3D-Architektur beschrieben wurde, soll in diesem Abschnitt näher auf die Möglichkeiten eines Sicherheitsmanagements für 3D Dokumente eingegangen werden. Um mit 3D Dokumenten in offenen Informationsräumen arbeiten zu können, müssen Methoden zum Copyright- und Vertraulichkeitsschutz innerhalb einer Software-Architektur realisiert werden. Dies geschieht hier, indem der variable Informationsgehalt der Objektrepräsentationen zum Schutz von Originaldokumenten verwendet wird. Dazu werden Document-Distribution-Classes und Benutzertypen festgelegt, die eine Kontrolle der Verteilung von Kopien erlauben. Nach der Diskussion von geeigneten Methoden zum Sicherheitsmanagement, die mit Hilfe des Document-Request-Brokers implementiert wurden, werden dann praktische Ergebnisse beschrieben.

8.4.1 *Document-Distribution-Classes* für 3D Dokumente

Zur Formulierung von Sicherheitsinformationen, hier als IPR-Info bezeichnet, werden verschiedene Document-Distribution-Classes (DDC) definiert, die festlegen, welche Übergänge von einer Repräsentationsform in eine andere für ein 3D Dokument (global) bzw. die enthaltenen Objekte (lokal) erlaubt sind. Diese Information ist neben den 3D Dokumenten in der Datenbank des Informations-Servers gespeichert. Zusammen mit der DDC-Information können Metadaten (Autorenname, Entstehungsdatum etc.), Daten über Dokument- bzw. Objekttypen und eine Wert mC für die maximale Auflösung von Kopien in der Datenbank dem Dokument zugeordnet sein. Beim Einlesen eines Dokumentes werden die IPR-Infos aus der Datenbank ausgelesen und in den Szenengraphen als Datenfeld eines Label-Knotens integriert (vgl. Abbildung 6.11). Dazu werden alle benötigten Informationen in einer festgelegten Syntax in einer Textkette zusammengefügt, wie es das folgende Beispiel zeigt:

“Metadata:Author=...,Date=XX.YY.ZZ; DDC:DDC-A,mC=0.5; oivType=Sphere“

Bei der Selektion eines Objektes wird die Information des IPR-Label-Knotens in der Untergruppe, in der sich das Objekt befindet, ausgelesen und mit den Parametern der Benutzerinformation verglichen. Aus dem Vergleich wird dann ermittelt, welche Bereiche des LoI-Reglers für den Zugriff gesperrt werden. Bei der Aktivierung wird die IPR-Information in ein internes Datenfeld des Interface-Objektes eingetragen und kann dort direkt weiterverwendet werden.

Es werden vier Document-Distribution-Classes definiert, nämlich DDC-A, DDC-B, DDC-C und DDC-D. In Abbildung 8.13 sind die erlaubten Übergänge für die verschiedenen Document-Distribution-Classes aufgelistet. Dabei sind jeweils auch Wechsel zu Darstellungsformen mit niedrigerem Informationsgehalt erlaubt, also für $R_i \rightarrow R_{i-1}$ sind auch die Übergänge $R_i \rightarrow R_{i-2}$ und $R_i \rightarrow R_{i-3}$ zugelassen. Falls R_{i-j} nicht existiert (also für $i - j < 1$) oder nicht erzeugt werden kann, wird dieser Übergang $R_i \rightarrow R_{i-j}$ nicht ausgewertet.

	DDC-A	DDC-B	DDC-C	DDC-D
Übergänge	Ri->Ri	Ri->Ri-1	Ri->Ri-2	Ri->Ri-3

Abbildung 8.13: Erlaubte Übergänge für verschiedene Document-Distribution-Classes.

Wenn also ein Objekt mit DDC-A gekennzeichnet ist, darf ein autorisierter Benutzer Kopien des Originals erhalten. Ob das Objekt in parametrischer Darstellungsform oder als 3D Netz vorliegt, ist dabei ersteinmal nicht ausschlaggebend, d.h. für ein 3D Netz mit DDC-A dürfen Kopien als 3D Netz mit voller Auflösung verteilt werden.

8.4.2 Festlegung von Benutzertypen

Für das Anwendungsszenario werden Benutzer mit unterschiedlichen Zugriffsrechten definiert. Im einfachsten Fall existieren vier Benutzertypen, nämlich *Gast, Mitglied, Administrator* bzw. *Besitzer*. Daneben sollten aber auch Benutzer mit individuellen Zugriffsrechten zugelassen und unterstützt werden. Gemeint sind hier Benutzer, die mit einem DL-Server zum Beispiel einmalig als Gast beim Durchsuchen verschiedener Informationsquellen arbeiten oder als Mitglied eines Dienstes eingetragen sind. Durch Erwerb von Rechten können Benutzer auch Originalkopien von Dokumenten erhalten. Der Administrator ist für die Pflege des Systems und die Inhalte verantwortlich und hat dafür freien Zugriff auf alle Komponenten und Dokumente.

Die Benutzertypen sind durch Attribute gekennzeichnet, die die maximal erlaubte Darstellungsform von Objektkopien in der Client-Szene widerspiegeln. Durch Angabe einer maximalen DDC und eines maximalen Komplexitätswertes ergeben sich die Regeln, nach denen Kopien als Objektrepräsentationen verteilt werden dürfen. In Abbildung 8.14 sind für die oben genannten Benutzertypen diese Bedingungen angegeben.

	Gast	Mitglied	Administator/Eigentümer
max. DDC	DDC-A	DDC-A	freier Zugriff
max. Komplexität	0.00	0.80	1.00

Abbildung 8.14: Benutzertypen und ihre maximalen Zugriffsrechte.

Hier besitzen Gäste und Mitglieder das Recht, Objekte mit dem maximalen Attribut DDC-A zu erhalten. Dies wird jedoch eingeschränkt durch die Angabe der maximalen Komplexität, die für Gäste bei $mC = 0.0$ bzw. für Mitglieder bei $mC = 0.8$ liegt. Dadurch bekommen Gäste während des Arbeitens mit 3D Dokumenten lediglich die Bounding-Box-Information von Objekten, mit denen sie lokal interagieren können. Mitglieder können dagegen mit relativ hoch aufgelösten 3D Netzen arbeiten. Für Eigentümer bzw. einen Administrator wird der Zugriff auf das Original zugelassen.

8.4.3 Vergleich von Dokumentinformationen und Benutzerinformationen

Die IPR-Informationen, die beim Laden eines 3D Dokumentes aus der Datenbank in den SCA3D-Server übernommen werden, werden beim Zugriff eines Benutzers auf die Objekte ausgewertet, um die aktuelle Rechtesituation zu klären. Dazu werden die bei der Benutzeranfrage mitgeschickten Benutzerinformationen mit den IPR-Informationen verglichen und anschließend die erlaubten Bereiche des LoI-Schiebereglers freigegeben. Positionen des LoI-Reglers, die beim Aktivieren eines Objektes die Rechte verletzen würden, werden für den Benutzer gesperrt. In Abbildung 8.15

sind die Informationen auf Server-Seite und auf Benutzerseite gezeigt, die für das Sicherheitsmanagement verwendet werden.

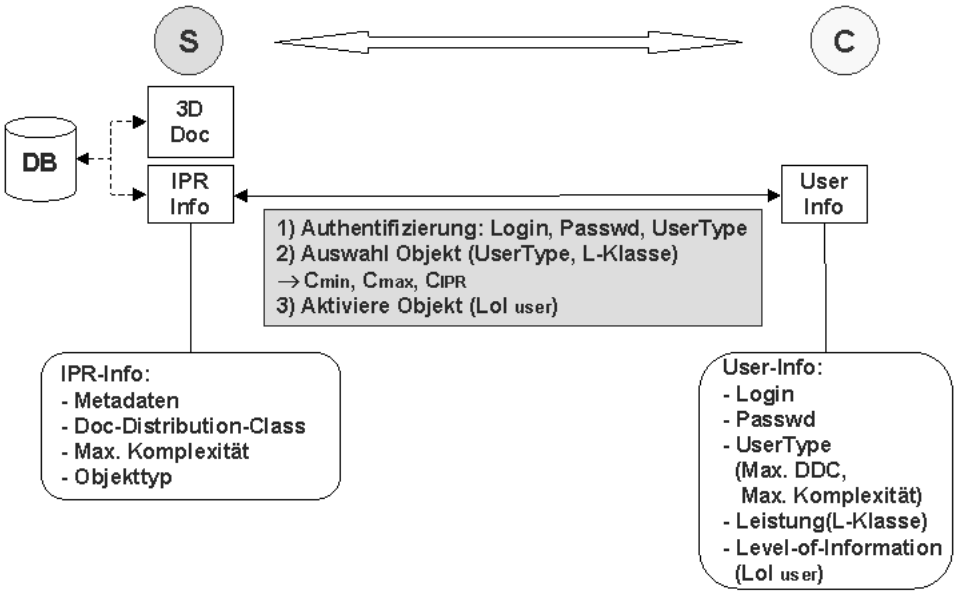

Abbildung 8.15: Vergleich von Benutzerinformation und Dokumentinformation.

Der Zugriff auf Dokumente ist in drei Schritte gegliedert, die auch in Abbildung 8.15 wiedergegeben sind. Ein Benutzer muß sich durch Senden seiner Kennung und eines Passwortes am Server authentifizieren. Die Login-Informationen verschiedener Benutzer werden dabei am Server verwaltet. Nach der Authentifizierung, die automatisch bei jedem Zugriff vorgenommen wird, kann der Benutzer das gewünschte Objekt auswählen. Bei der Selektion wird der Benutzertyp und die L-Klasse der Client-Anwendung an den Server als Parameter eines Remote-Document-Call geschickt. Aus diesen Parametern und den Informationen über das Dokument am Server werden dann die Werte c_{min}, c_{max} und der Wert c_{IPR}, der den gesperrten Bereich des LoI-Reglers festlegt, bestimmt und zur Client-Anwendung übertragen. Dort wird der LoI-Regler bzw. das entsprechende Panel-GUI entsprechend eingestellt, sodaß der Benutzer Informationen über die optimalen Darstellungsform des gewählten Objektes und die gesperrten Darstellungsformen erhält. Im dritten Schritt wählt der Benutzer eine eigene LoI-Position aus, die innerhalb des erlaubten Bereiches liegen muß und innerhalb des optimalen Bereiches zwischen c_{min} und c_{max} liegen sollte, und aktiviert das Objekt. Daraufhin wird die passende Objektrepräsentation in die Client-Szene integriert.

Beim Abgleich von IPR-Information und User-Information wird in jedem Fall die server-seitige Information, also Angaben über DDC und maximale Auflösung, höher bewertet. Ein Dokument, das mit einem Attribut DDC-B versehen ist und im Original als 3D Netz vorliegt, wird daher immer nur als Bounding-Box-Darstellung übertragen. Auch wenn der Benutzer durch seine L-Klasse und seinen Benutzertyp einen höheren Level-of-Information erhalten könnte, ist die Sicherheitsinformation ausschlaggebend für die tatsächlich verteilten Objektkopien. D.h. die Mechanismen des Sicherheitsmanagements greifen immer vor den Mechanismen des Komplexitätsmanagements. Nur wenn eine Benutzer die Rechte am Original erworben hat, werden die Originaldaten als Kopie versendet, wobei sie mit einer speziellen Sicherheitsinformation für den Besitzer versehen werden.

Der Einsatz von digitalen Signaturen und Wasserzeichen (siehe [BB00] und [Dit00]) ist dabei ein wirksames Mittel, um die Verteilung von Originalkopien zu kontrollieren.

8.4.4 Bereitstellung geeigneter Methoden mit dem Document-Request-Broker

Zur Realisierung des Sicherheitsmanagements werden Methoden verwendet, die mit Hilfe des Document-Request-Brokers vom Client aus aufgerufen werden. Das Interface-Objekt eines aktivierten Dokumentobjektes bietet diese Methoden für den entfernten Aufruf an und erlaubt den kontrollierten Zugriff auf die Sicherheitsinformationen und die geschützten Daten. Für das Anwendungsszenario dieses Kapitels werden zwei einfache Methoden implementiert, die ein Benutzer zum Zugriff auf das Dokument verwenden kann.

Um die Schutzinformationen eines Dokumentes bzw. eines Objektes direkt abzufragen, wird die Methode *getIPRInfo()* benutzt, die den IPR-String des selektierten Objektes zurückliefert. Die Art der Weiterverarbeitung der Information wird der Client-Anwendung überlassen und ist nicht geregelt. Diese Methode unterstützt den Punkt Evaluation von Sicherheitsinformationen, der in Abbildung 2.7 des Kapitels 2 für das dort vorgeschlagenen Dokumentmodell beschrieben wurde. Ein Benutzer kann auf diese Weise schon vor der Aktivierung eines Dokumentobjektes Informationen über die Schutzrechte, wie z.B. den Eigentümer und Copyright-Daten, oder den Dokumenttyp, wie z.B. parametrisches Modell oder 3D Netz, erfahren. Dadurch kann schon vor einer Übertragung der eigentlichen Dokumentdaten entschieden werden, welche Schritte zum weiteren Arbeiten mit dem Dokument unternommen werden sollten. In der Praxis könnte z.B. ein Planer, der 3D Dokumente mit einer bestimmten Auflösung benötigt, als Gast in einer Produktdatenbank einer Anbieterfirma suchen, wobei die interaktive Visualisierung über entfernt gerenderte Bilddaten ermöglicht wird. Wenn ein passendes 3D Objekt gefunden wurde, kann anhand der IPR-Information vorab geklärt werden, über welche Kontakte und mit welchen Verfahren eine hochaufgelöste Kopie erworben werden kann.

Die zweite Basismethode, die für das Sicherheitsmanagement implementiert wurde, ist die Methode *getOriginal()*. Diese Methode ist eine spezialisierte Form der Methode $getOR()$ des Interface-Objektes. Wenn ein Benutzer die Rechte an einem Originaldokument erworben hat, wird ihm diese Methode von seiner Client-Anwendung bereitgestellt. Eine Benutzer, der nicht zuvor die Rechte zur Kopie eines Originals erworben hat, kann diese Methode *getOriginal()* nicht über seine Anwendung aufrufen, da sie gesperrt bleibt. Durch Selektion eines Objektes und Remote-Aufruf dieser Methode werden die Originaldaten vom Server zum Client übertragen und der Benutzer kann sie dort lokal abspeichern. Dabei werden die IPR-Informationen, die im Allgemeinen eine Originalkopie verhindert hätten, für diesen einen Vorgang außer Kraft gesetzt. Das Original ist dabei als Originalkopie zu verstehen, d.h. das Dokument bleibt in der Server-Datenbank weiter mit den vorher gültigen Schutzrechten bestehen.

Weitere Methoden zum Sicherheitsmanagement sind denkbar. Zur Kennzeichnung von Kopien könnten Integrationsmethoden für Wasserzeichen und digitale Signaturen im Interface-Objekt implementiert werden. Um eine Dokumentverwaltung zu ermöglichen, wären auch Abfragemethoden zum Bearbeitungsstand und zu den bearbeitenden Personen sinnvoll. Schließlich könnte durch Anbieten von speziellen Methoden, wie z.B. *getEncryptedObject()*, die verschlüsselte Übertragung mit Hilfe von SSL-Verbindungen (*Secure Socket Layer* [FKK96]) erreicht werden.

8.4.5 Praktische Ergebnisse bei der Anwendung eines abgestuften Rechtekonzepts

Das Sicherheitsmanagement innerhalb der SCA3D-Architektur soll nun anhand des praktischen Beispiels aus Abschnitt 8.3.4 erläutert werden. Bei der Selektion verschiedener 3D Objekte innerhalb eines geladenen 3D Dokumentes wurden dort die Schranken der Komplexitätswerte c_{min} und c_{max} bestimmt und daraus die optimale Darstellungsform für das Objekt am Client ermittelt. Dabei waren nur Performanzkriterien von Bedeutung. Nun sollen die dort gefundenen Ergebnisse dahingehend erweitert werden, daß auch die Auswirkungen des Sicherheitsmanagements erkennbar werden. Dazu werden für zwei Benutzer (Mitglied und Gast) und ein einfaches Beispieldokument die Benutzerinformation und die IPR-Info in die Auswertung miteinbezogen. Die Ergebnisse sind in Abbildung 8.16 gezeigt.

	L-Klasse	Sphere (R=2.0)	Sphere (R=2.0)
Doc-Info		DDC: DDC-B; mC=0.7; Type: OIVSphere	DDC: DDC-B; mC=0.7; Type: OIVSphere
User-Info		Login= XYZ, PassWd=xxx; User-Type: **Member**; L-Class:L2	Login= ABC, PassWd=yyy; User-Type: **Gast**; L-Class:L2
Client 2	L2	Cmin = 0.6	Cmin = 0.6
		Cmax = 0.8	Cmax = 0.8
		C_{IPR} = 0.7	C_{IPR} = 0.0
		Erlaubte Übergange: R1->R2	**Erlaubte Übergange: R1->R3**
		LoI user = 0.7 (3D Netz)	**LoI user = 0.0 (Bounding-Box)**
Objektrepräsentation		**3D Netz, 1568 Dreiecke, ~150kByte**	**Bounding-Box, 2 Eckpunkte, < 100 Byte**

Abbildung 8.16: Auswahl von Objektrepräsentationen mit Sicherheitsmanagement.

Das 3D Dokument ist mit den Attributen DDC-B und $mC = 0.5$ belegt, darf also nur als 3D Netz mit der angegebenen maximalen Komplexität übertragen werden. Der Typ ist mit $OIV\,Sphere$ gekennzeichnet, d.h es handelt sich um ein parametrisches OpenInventor-Kugelobjekt. Die Benutzerinformation gibt neben den Login-Daten auch den Benutzertyp (Mitglied bzw. Gast) und die L-Klasse der Client-Anwendung (L2) an. Außerhalb dieser Rechtesituation könnte der Benutzer sowohl das Original als auch 3D Netze mit einer Auflösung zwischen $c_{min} = 0.6$ und $c_{max} = 0.8$ anfordern. Diese Ergebnisse waren in Abschnitt 8.3.4 ermittelt worden. Mit den Einschränkungen durch die Document-Distribution-Class und die maximale Auflösung des 3D Dokumentes darf das Mitglied nun bei aktiviertem Sicherheitsmanagement nur 3D Netze mit einer maximalen Komplexität von $c_{IPR} = 0.7$ erhalten, da Originalkopien durch das DDC-Attribut nicht zugelassen sind. Der Benutzer, der als Gast mit dem Informationssystem arbeitet, darf im Gegensatz dazu nur mit der Bounding-Box-Information der Objekte $c_{IPR} = 0.0$ lokal arbeiten, während er vom Server natürlich genauso mit Bilddaten versorgt wird wie das eingetragene Mitglied.

Dem Benutzer wird am Panel-GUI angezeigt, welche Bereiche des LoI-Reglers empfohlen und welche gesperrt sind. Innerhalb des empfohlenen Bereiches ist noch eine Wahl zwischen verschiedenen Auflösungen möglich, so daß der Benutzer eine eigene LoI-Position einstellen kann. Die im Beispiel gewählte Position entspricht dem maximal erlaubten Komplexitätswert $LoI = c_{IPR} = 0.7$. Als Resultat wird ein Kugelobjekt mit Radius $R = 2.0$ (interne Längeneinheiten) als 3D Netz mit insgesamt 1568 Dreiecken in die Client-Szene integriert, wobei eine Datenmenge von ungefähr 150 MByte übertragen werden muß. In Abbildung 8.17 ist ein praktisches Beispiel anhand des

SCA3D-Prototypen gezeigt. Einem Benutzer werden je nach seiner Rechtesituation verschiedene Objektrepräsentationen lokal zur Verfügung gestellt. In diesem Fall ist dies entweder die Bounding-Box-Information im Navigationsraum, um ein Objekt z.B. besser lokalisieren und unterscheiden zu können, ein 3D Netz mit einem bestimmten Komplexitätswert oder das Original als NURBS-Fläche.

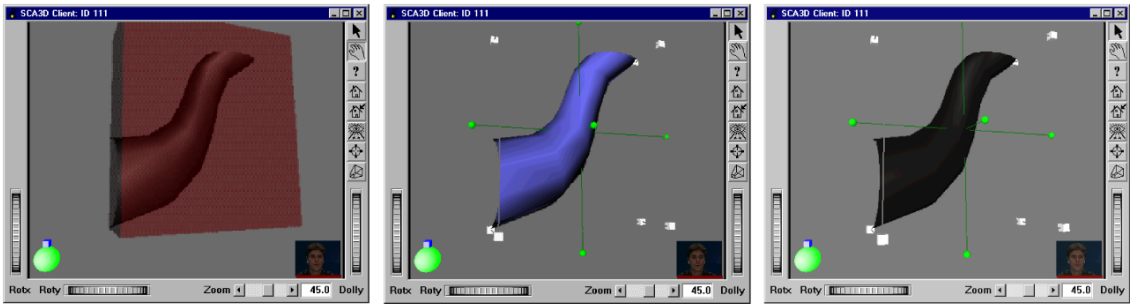

Abbildung 8.17: Verfügbare Objektrepräsentationen: Bounding-Box-Information, 3D Netz oder originale NURBS-Fläche.

Durch Auswahl geeigneter Einstellungen für die Attribute DDC und maximale Komplexität wird zusammen mit den Benutzerrollen ein abgestuftes Rechtekonzept eingeführt. Dabei ist es möglich, daß die Auflösung der Kopien auf Benutzerseite an die erworbenen Rechte angepaßt wird. In dem hier diskutierten Anwendungsszenario wurden nur die Benutzerrollen Gast, Mitglied und Administrator bzw. Besitzer untersucht. Für ein erweitertes Szenario mit individuellen Rechten kann dieses Konzept aber mit einer verfeinerten Abstufung umgesetzt werden und ermöglicht damit ein flexibles System zur Rechtevergabe in offenen Informationsräumen.

8.5 Zusammenfassung

In diesem Kapitel wurden anhand eines Anwendungsszenarios die Anforderungen beim Arbeiten mit 3D Dokumenten in offenen Informationsräumen diskutiert. Also Anwendungsumfeld wurden dafür verteilte Digitale Bibliotheken gewählt, die in ihren Eigenschaften den offenen Informationsräumen nahe kommen. Insbesondere Techniken zum Umgang mit komplexen und geschützten 3D Dokumenten wurden dabei als notwendige Voraussetzung für ein anwendungsorientiertes System erkannt. Die SCA3D-Software-Architektur wurde dann im Hinblick auf ihr Potential für ein Komplexitätsmanagement und ein Sicherheitsmanagement in heterogenen verteilten Umgebungen untersucht und Umsetzungsmöglichkeiten beschrieben. Durch Klassifizierung von DL-Dokumenten bezüglich ihres Informationsgehaltes und der Einordnung von Client-Anwendungen in Leistungsklassen wurde das Komplexitätsmanagement realisiert und Messungen in einer Testumgebung vorgestellt, die die Wirksamkeit des Konzeptes belegen.

Für die Realisierung des geforderten Sicherheitsmanagements wurden Document-Distribution-Classes eingeführt, die eine Kontrolle der Verteilung von Kopien erlauben. Mit Hilfe von Benutzertypen in einem Rollenmodell wurde der Zugriff auf die so klassifizierten Dokumente reguliert und

an die aktuelle Rechtesituation angepaßt. Mit Hilfe spezieller Methoden des Document-Request-Brokers der SCA3D-Architektur wurden dem Benutzer Möglichkeiten zum Umgang mit geschützten Dokumenten zur Verfügung gestellt, wobei die Funktionalität durch Implementierung weiterer Methoden jederzeit auszudehnen ist. Praktische Ergebnisse beim Arbeiten mit 3D Dokumenten bei aktivem Sicherheitsmanagement wurden vorgestellt und die Flexibilität bei der Umsetzung von Rechtekonzepten hervorgehoben.

Kapitel 9

Kooperatives Arbeiten innerhalb der SCA3D Architektur

In diesem Kapitel wird die Software-Architektur im Hinblick auf ihre Einsetzbarkeit und ihre Vorteile für kooperative Anwendungen untersucht. Damit sind solche Anwendungen gemeint, die mehreren Benutzern erlauben, synchronisiert mit den Inhalten von 3D Dokumenten zu arbeiten, wobei sie durch verteilte 3D Visualisierung unterstützt werden. Ansätze für verteilte Visualisierungssysteme für mehrere Benutzer wurden bereits in Kapitel 4 beschrieben. Hier soll es vorallem darum gehen, die neuen Möglichkeiten der SCA3D-Architektur beim Arbeiten mit 3D Dokumenten in solchen Mehrbenutzerumgebungen zu untersuchen. Besonders die Vorteile im Umgang mit komplexen und geschützten Dokumenten in heterogenen Umgebungen, d.h. solchen mit Unterschieden im Hinblick auf Bandbreite und lokale Leistung zwischen den Benutzerrechnern, werden besprochen. Mit der dynamischen Kombination von lokaler und Remote-Visualisierung deckt die SCA3D-Architektur das Spektrum zwischen Systemen, die nur mit lokaler Visualisierung und Datenreplikation arbeiten, und reinen Remote-Visualisierungssystemen ab. Es soll untersucht werden, auf welche Weise sich die wichtigsten Anforderungen an eine verteilte 3D Visualisierungsumgebung für mehrere Benutzer durch diesen kombinierten Ansatz realisieren lassen.

9.1 Einführung: Anforderungen an die Visualisierungsumgebung bei der Kooperation

Die wichtigsten Anforderungen an Mehrbenutzersysteme zur verteilten 3D Visualisierung werden im folgenden beschrieben, wobei Basisanforderungen und fortgeschrittene Anforderungen unterschieden werden. Diehl nennt in [Die01] als Basisanforderungen an eine virtuelle Mehrbenutzerwelt:

- *Objekte hinzufügen und entfernen*: Wenn Benutzer sich anmelden oder abmelden bzw. Objekte in ein Dokument einfügen oder entfernen, sollen alle anderen Benutzer-Szenen diese Veränderungen wiederspiegeln. Dazu müssen Szenenveränderungen durch Objektübertragungen und -integration möglich sein und Objekte und Benutzer müssen registriert werden.
- *Kommunikation von veränderten Objektzuständen*: Objekte werden durch eine graphische Repräsentation, durch einen Zustand und ein Verhalten beschrieben. Alle Änderungen am Zustand der Objekte, z.B. ihre Position, Ausrichtung, Materialeigenschaften etc., sollen allen

Benutzern möglichst zum gleichen Zeitpunkt sichtbar gemacht werden. Wenn das Verhalten sich ändert, z.B. weil die Zugriffsrechte eingeschränkt werden, muß sich dies auch auf alle Benutzeranwendungen auswirken. Oft wird die Anwesenheit der Benutzer durch die Positionen ihrer Kameras und des Blickwinkels den anderen Teilnehmer sichtbar gemacht. Die Objekte in den verteilten Szenen, die einen Benutzer vertreten, werden als *Avatare* bezeichnet. Auch die Zustandsänderung von Avataren müssen immer kommuniziert und aktualisiert werden.

- *Benutzerkommunikation durch Audio/Video-Streaming*: Durch Audio- bzw. Videokonferenzen soll die Kooperation zwischen den Benutzern erleichtert werden. Dazu wird in der Regel AV-Streaming verwendet, um eine unterbrechungsfreie Kommunikation zu ermöglichen.

Als fortgeschrittene Anforderungen an Mehrbenutzersysteme werden vorallem die beiden Aspekte der Adaptivität an die Randbedingungen während des Arbeitens und des Sicherheitsmanagements für Daten- und Rechnerzugriff angesehen.

- *Adaptivität an die Randbedingungen*: Eine Client-Anwendung, die sich an einer kooperativen Visualisierungskonferenz beteiligt, sollte durch die übertragenen Daten und lokal ablaufenden Prozesse nicht überlastet werden. Die zu übertragende und lokal zu bearbeitende Informationsmenge sollte dynamisch so angepaßt werden, daß die Verzögerung zwischen allen Teilnehmern in einem festgelegten Rahmen bleibt.

- *Sicherheit von Daten und beteiligten Ressourcen*: Die Bearbeitung und Verteilung von 3D Dokumenten sollte in einer Weise kontrollierbar sein, daß kein unerlaubter Zugriff, unerwünschte Datenmanipulation oder Verletzung von anderen Rechten möglich ist. Der Zugriff auf Rechner über das Netzwerk sollte nicht zu Sicherheitslücken und damit zur Gefährdung der beteiligten Rechner führen.

Bei der Realisierung dieser Anforderungen sind folgende Punkte zu beachten. Erstens werden in einem Netzwerk eventuell nur niedrige Bandbreiten mit einer hohen Verzögerungszeit zur Verfügung stehen. Auf jeden Fall werden diese Parameter für verschiedene Benutzer variieren. Zweitens ist die Heterogenität des Netzwerkes bezüglich der verwendeten Technologie, wie Betriebssystemen und vorhandenen Programmbibliotheken, zu berücksichtigen. Drittens müssen die Prozesse, die für die verteilte Interaktion lokal ablaufen, im gesamten Rechnernetz synchronisiert werden. Viertens sollten Möglichkeiten zum Fehlerausgleich eingeführt werden. Fünftens schließlich sollte eine solche Umgebung mit der Anzahl ihrer Benutzer skalieren, d.h. bei steigender Anzahl von Benutzer sollte das Verhalten des Systems sich nicht zu stark ändern. Insbesondere sollten die Verzögerungzeiten beim Arbeiten mit Objekten nicht unverhältnismäßig ansteigen.

9.2 Kooperative Szenarien unter Verwendung des SCA3D Konzeptes

Mit Hilfe der prototypischen Implementierung der SCA3D-Architektur wird ein Mehrbenutzer-Szenario umgesetzt. Die Aufteilung in einen Navigationsbereich und einen lokalen Interaktionsbereich soll hier im Hinblick auf die verbesserte Unterstützung kooperativen Arbeitens untersucht werden. In Abschnitt 5.7 wurde bereits das kooperative Modell der SCA3D-Architektur, das mit Hilfe eines CORBA-basierten Ansatzes arbeitet, vorgestellt. Nun sollen die Anwendungsmöglichkeiten, die durch diesen technischen Ansatz realisierbar sind, ausgewertet werden. Bei dem hier verwendeten Kooperationsmodell handelt es sich, um ein Client-Server-Modell mit

Unicast-Kommunikation. Andere Ansätze wie Peer-to-Peer-Konzepte (siehe z.B. das DIVE-System [CH93]) werden hier nicht untersucht, da das Client-Server-Modell für den Anwendungsbereich offener Informationsräume bzw. Digitaler Bibliotheken mit Informations-Servern vorgegeben und vorteilhafter ist. Multicast-Kommunikation könnte z.B. zur Synchronisation von gleichzeitig aktivierten Objekten benutzt werden, wobei alle Benutzer eine Multicast-Gruppe bilden, die das gleiche Objekt aktiviert haben.

Das Konzept von Navigations- und Interaktionsbereichen kann vorteilhaft sowohl im Hinblick auf adaptive 3D Visualisierung als auch für das Sicherheitsmanagement bei der Kooperation zwischen Benutzern angewendet werden. Die Kooperation in heterogenen Netzen mit Benutzern, die sehr unterschiedliche Endgeräte und Anbindungsbandbreiten verwenden, wird damit möglich. Als Beispiel sei die Zusammenarbeit eines Projektleiters auf einer Baustelle mit PDA und mobiler Anbindung und einem Ingenieur im Büro einer beauftragten Firma mit einem leistungsstarken PC und einer guten Anbindung genannt. Die Bearbeitung eines Änderungswunsches an einem Bauteil kann mit dem hier vorgestellten Ansatz in diesem Praxisfall ermöglicht werden, wobei auch die Rechtesituation für geschützte technische Informationen berücksichtigt werden kann.

9.2.1 Nutzung von Navigations- und Interaktionsbereichen

Da in dem beschriebenen Szenario die graphischen Repräsentationsformen von 3D Objekten in den Client-Szenen variable und dynamisch verteilt werden können, kann die Verzögerungszeit zwischen den Benutzern durch Anpassung der Client-Szenen minimiert werden. Ein Benutzer mit einer schlechten Performanz arbeitet dann mit Repräsentationsformen, die die verfügbaren Ressourcen schonen, während eine Benutzer mit guter Anbindung und lokaler Leistung mit hochaufgelösten Objektrepräsentationen in seiner Client-Szene arbeiten kann. Die Rechtesituation führt zu einer Kontrolle der Repräsentationsformen in den Client-Szenen. Auch wenn der schwache Client die Möglichkeit nutzt, hauptsächlich lokal zu arbeiten und das Server-Feedback zu verringern, werden alle Aktionen über den Rückkanal sofort zum Server gesendet. Der starke Client kann diese Aktionen nahezu in Echtzeit sehen, da er mit einem schnellen Server-Feedback versorgt wird. Die Verzögerung vom Client zum Server ist also immer gering, während die Verzögerung vom Server zum Client über das visuelle Feedback reguliert werden kann. Falls der schwache Client in diesem Szenario sich in einer Phase befindet, in der er lokal nicht mit Objekten interagieren muß, kann er das Server-Feedback auf einen maximalen Wert regeln, um die Aktionen der anderen Benutzer möglichst sofort zu sehen.

In Abbildung 9.1 ist eine Szenario mit zwei Benutzern gezeigt, die innerhalb eines 3D Dokumentes auf Server-Seite navigieren und gleichzeitig Objekte in ihrem jeweiligen Interaktionsraum lokal bearbeiten. Der Interaktionsraum kann dabei auf Server-Seite mit einem räumlichen Bereich verknüpft werden, so daß bei der Bewegung eines Benutzers in der Szene die Objekte in diesem Bereich aktiviert werden. Wenn sich die Interaktionsräume in diesem Szenario überschneiden, können die Benutzer gleichzeitig lokal auf Objekte zugreifen. Durch Festlegung von Regeln zum gleichzeitigen Zugriff muß eine Interessenkollision vermieden werden. Die einfachste Weise zur Regelung besteht hier in der Vergabe eines Tokens, der das alleinige Zugriffsrecht auf ein gleichzeitig aktiviertes Objekt für einen Zeitabschnitt einem Benutzer zuordnet. Auf Anfrage wird der Token dann weitergegeben bzw. falls längere Zeit keine Interaktion stattfindet wird der Token automatisch freigegeben.

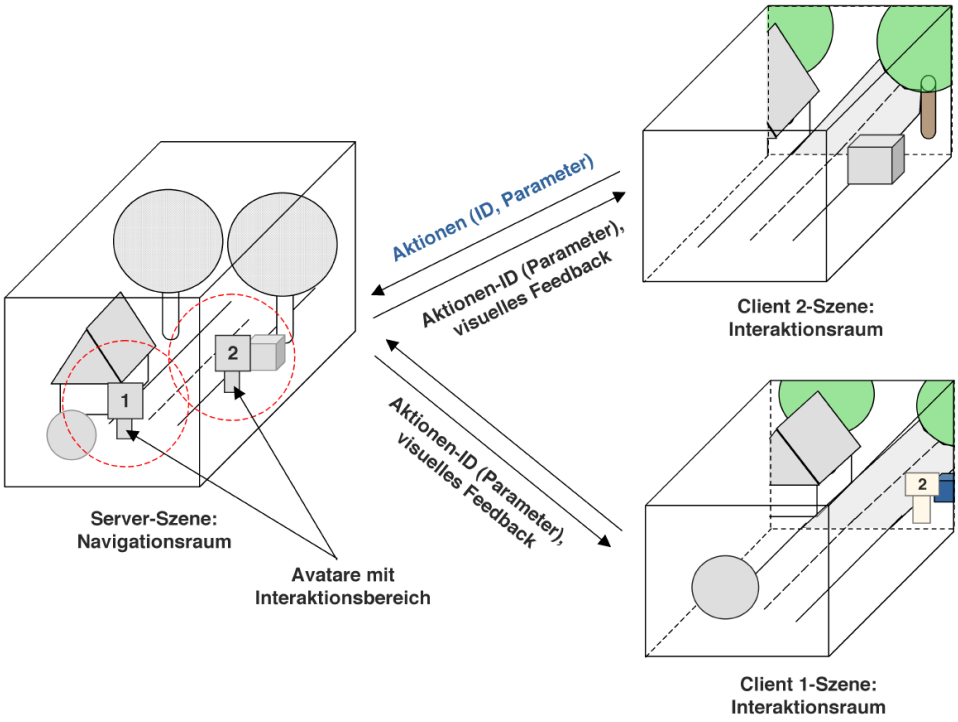

Abbildung 9.1: Kooperatives Szenario innerhalb der SCA3D-Architektur.

Synchronisation der Navigation

Um einen kooperativen visuellen Eindruck bei den Benutzern zu erzeugen, werden verschiedene Techniken zur Synchronisation von Blickrichtung- und position (Kamerasynchronisation) und zur Repräsentation der Benutzer in der Szene durch Avatare eingesetzt. In diesem Abschnitt soll die Kamerasynchronisation näher beschrieben werden. In der SCA3D-Architektur sind zwei Arten der Kamerasynchronisation realisiert, nämlich einerseits gemeinsame Navigation und andererseits unabhängige Navigation. Gemeinsame Navigation bietet sich zum Beispiel bei einem Tutor-Klasse-Szenario an, bei dem eine Gruppe von Schülern von einem Tutor durch ein virtuelles Modell geführt wird. Für andere kooperative Anwendungen, wie z.B. dem gemeinsamen Bearbeiten eines Konstruktionsbauteils, wird in den meisten Fällen eine unabhängige Navigation stattfinden.

In Abbildung 9.2 sind die Abläufe bei der Navigationssynchronisation verdeutlicht, wobei ein aktiver und ein empfangender Client dargestellt sind. Die Synchronisation der Kameras erfolgt dabei immer über den Server, der die Kameraparameter weiterverteilt und gleichzeitig ein visuelles Feedback in Form von Bilddaten an die Client-Anwendungen sendet. Die Bilddaten werden auf Client-Seite auf die Projektionsfläche abgebildet, die sich mit der Kamera bewegt und auch ihre Orientierung vor der Kamera beibehält. Die Übertragung der Aktionen erfolgt dabei in beide Richtungen möglichst schnell (CORBA RPCs), während die Geschwindigkeit des visuellen Feedbacks über die Remote-Framerate geregelt werden kann. Wenn ein Benutzer eine Aktion zum Server sendet wird seine Kennung (ID) als Übergabeparameter mitgeliefert, sodaß der Server die Client-Anwendung identifizieren kann. Wenn der Server die Aktion weiterleiten soll, wird die Kennung der zu rufenden Client-Anwendung an den Aktionsnamen angehängt, da die CORBA-Referenzen

der einzelnen Benutzer nach diesem Schema benannt werden.

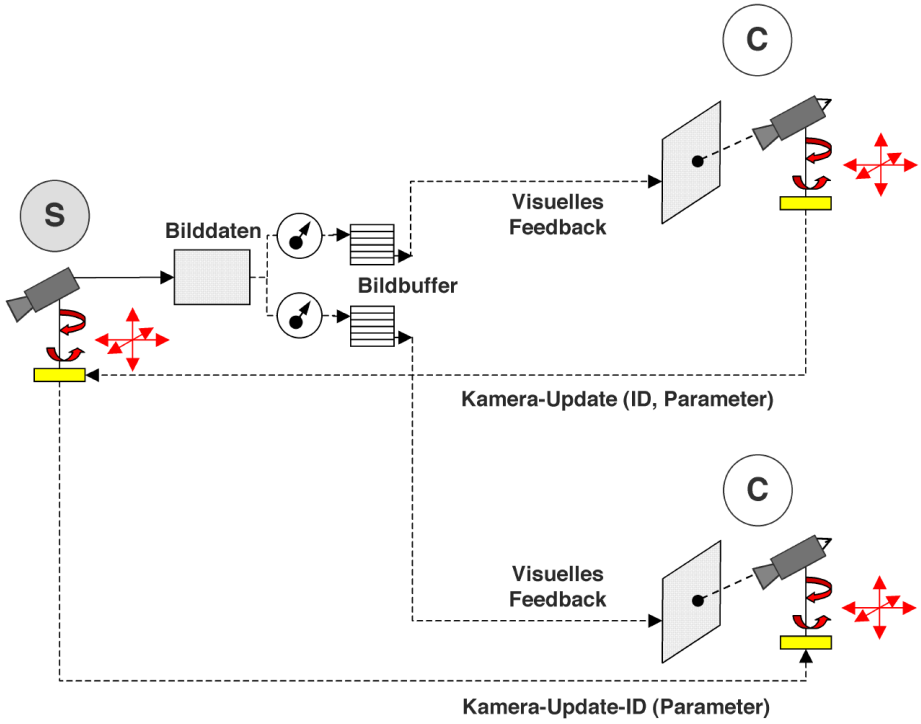

Abbildung 9.2: Synchronisation der Kameras und des visuellen Feedbacks.

In Abbildung 9.1 sind auch Avatare für die beiden Benutzer als einfache 3D Objekte in die Server-Szene integriert worden. Ihr Interaktionsbereich wird hier durch einen kugelförmigen Raum mit der Kameraposition als Mittelpunkt repräsentiert. Wenn ein Benutzer im 3D Dokument navigiert, werden die anderen Benutzer über das visuelle Feedback darüber informiert, d.h. die Avatare befinden sich hier auf Server-Seite und müssen nicht wie bei anderen Systemen in den Client-Szenen repräsentiert werden.

- *Gemeinsame Navigation*: Bei der gemeinsamen Navigation werden alle Parameter von Client-Kameras und Server-Kamera zu jedem Zeitpunkt synchronisiert. Wenn ein Benutzer seinen Blickpunkt ändert, werden diese neuen Parameter zum Server gesendet und dort auf die Kamera angewendet. Gleichzeitig werden die Parameter an alle anderen Benutzer gesendet. Hierbei ist darauf zu achten, daß die ursprünglich sendende Client-Anwendung nicht synchronisiert wird, da sonst Schleifeneffekte auftreten, die das System blockieren können. Die Kameraparameter dienen dazu, die Position, die Orientierung und den Fokus der Kamera einzustellen. Wie in Abbildung 9.2 durch Pfeile angedeutet angedeutet, besitzt eine Kamera alle sechs Freiheitsgrade der Bewegung, d.h. Translation entlang der drei Grundachsen und Rotation um diese Achsen. Ein Benutzer der das Recht dazu hat, kann für alle Benutzer auf gemeinsame Navigation umschalten, d.h. ein Tutor könnte auf diese Weise für einen Zeitraum die Kameraführung übernehmen.

- *Unabhängige Navigation*: Wenn nun die Benutzer eigenständig navigieren wollten, wer-

den die Kameraparameter lediglich zwischen Client-Anwendung und Server synchronisiert. Die Kamerasynchronisation zwischen Server und Client wird also ausgeschaltet. Um jedem Benutzer das visuelle Feedback zu liefern, das seinen Kameraparametern entspricht, wird nun am Server in festgelegter Reihenfolge die 3D Szene mit den individuellen Kameraparametern gerendert. Die Bilddaten werden dann in den entsprechenden Bildbuffer des Benutzer geschrieben. Dabei können die verschiedenen Anforderungen der einzelnen Client-Anwendungen an das visuelle Feedback berücksichtigt werden, so daß leistungsfähige Client-Rechner auch in diesem Modus mit einer höheren Framerate versorgt werden können. Mit dieser Methode wird eine unabhängige Navigation der Benutzer ermöglicht, wobei allerdings die Leistungsfähigkeit des Servers entsprechend gut sein muß. Mit steigender Anzahl der Benutzer, die unabhängig navigieren wollen, wird entweder die Server-Last im Verhältnis der angeforderten Frameraten steigen, oder wenn der Server ausgelastet ist, werden die Benutzer mit einer verringerten Bildwiederholrate versorgt werden.

9.2.2 Unterstützung der Kooperation durch Informationsvisualisierung

Bei der Anforderungsanalyse in Abschnitt 6.2 wurden bereits die Bedienelemente in der Client-Szene angesprochen, die zur Visualisierung des Zustands der Mehrbenutzerumgebung verwendet werden. Diese dienen zur Informationsvisualisierung im kooperativen Arbeitsprozeß und gehören nicht zu dem eigentlichen 3D Dokument, das als Teil der Client-Szene angezeigt wird. Die wichtigsten Komponenten, die in die Client-Szene und die Server-Szene zu diesem Zweck integriert werden, sind Benutzerlogos, Navigationssignale und Avatar-Objekte. Die ersten beiden Elemente werden im Interaktionsraum eingeblendet, während die Avatar-Objekte in den Navigationsraum eingefügt werden. Die Benutzerlogos und die Navigationssignale werden gegenüber der sich bewegenden Kamera stets so ausgerichtet, daß sie auch bei der Navigation ortsfest im unteren Rand des Anwendungsfensters erscheinen.

- *Benutzerlogos*: Jedem Benutzer wird ein Logo zugeordnet, das als Polygon mit einer Fototextur in der Client-Szene realisiert wird. Das Foto des Benutzers wird lokal abgelegt und bei Bedarf eingelesen. Außerdem wird eine Farbstreifen an der unteren Kante des Logos eingeblendet, der für jeden Benutzer eine eigene Farbe festlegt, die zur Markierung von selektierten Objekten und Avataren benutzt wird. In Abbildung 7.7 sind Benutzerlogos für zwei kooperierende Benutzer gezeigt, denen jeweils auch eine Farbe zugeordnet wurde. Auch für den Benutzer der jeweiligen Client-Anwendung wird lokal das eigene Logo eingeblendet, um einen vollständigen Überblick der Benutzer zu bieten und die verknüpfte Farbe bekannt zu machen. Die Reihenfolge der Einblendungen der Logos ist in allen Client-Szenen so festgelegt, daß das eigene Logo in der rechten unteren Ecke des Client-GUI erscheint und weitere Benutzer dynamisch in der Reihenfolge ihrer Anmeldung links davon eingeblendet werden. Wenn ein Benutzer sich abmeldet, wird das Logo aus allen Client-Szenen entfernt und die Farbe freigegeben.

- *Navigationssignale*: Ebenfalls in Abbildung 7.7 ist in der linken unteren Ecke ein graphisches Objekt eingeblendet, das als Navigationssignal bezeichnet wird. Es dient dazu, dem Benutzer einerseits den aktuellen Navigationsmodus anzuzeigen und andererseits den Zustand der Objekte im Interaktionsraum zu signalisieren. Die zwei Elemente des Navigationssignals sind ein Kugelobjekt, das entweder grün oder rot gefärbt ist, und ein kleineres, auf die Kugel aufgesetztes Würfelobjekt, das entweder die Farbe Blau oder die Farbe Rot annimmt. Im einzelnen ist die Funktionalität wie folgt festgelegt: Bei gemeinsamer Navigation erscheint die

Kugel bei allen Benutzern grün, während unabhängige Navigation durch die Farbe Rot kenntlich gemacht wird. Das Umschalten zwischen den Navigationsmodi erfolgt über eine Tastenkombination. Mit Hilfe des Würfelobjektes wird angezeigt, ob die Verbindung zwischen dem selektierten server-seitigen Dokumentobjekt und der client-seitigen Objektrepräsentation vollständig aktiviert ist. Da meist mehrere Objekte im Interaktionsraum vorhanden sind, wird das aktuell bearbeitete Objekt am Server und am Client intern speziell markiert. Durch doppeltes Anklicken eines Client-Objektes wird dieses vollständig aktiviert. Dies wird durch die blaue Farbe des Würfelobjektes signalisiert, während eine nicht vollständige Verbindung bei der Interaktion mit einer roten Färbung angezeigt wird. In diesem Fall kann ein Objekt im Interaktionsraum z.B. verschoben werden, ohne das sich dies auf das Server-Dokument auswirkt. Ein Zustand, der im allgemeinen nicht gewünscht ist und der deshalb durch ein Warnsignal kenntlich gemacht wird.

- *Avatar-Objekte*: Die Anwesenheit von mehreren Benutzern bei der kooperativen Bearbeitung eines 3D Dokumentes kann durch Avatar-Objekte wahrnehmbar gemacht werden. Dazu werden festgelegte geometrische Formen, z.B. zwei farbige aufeinandergestellte Quader (siehe DIVE-System [CH93]), in die Szenen eingebracht. Diese Avatare sind jeweils mit einem Benutzer verknüpft, indem sie der Bewegung der Kamera dieses Benutzers folgen. Der Benutzer selbst sieht dann ein ortsfestes Objekt in seiner Szene, mit dem er sich eingeschränkt identifizieren kann, während andere Benutzer durch den Avatar die aktuelle Position des Benutzers erkennen können. In Systemen, die mit Datenreplikation und lokaler Visualisierung arbeiten, wird in jeder Client-Szene ein Avatar-Objekt für jeden Benutzer integriert. Alle diese Avatare müßen in ihren Bewegungen synchronisiert werden, wodurch bei einer steigenden Anzahl von Benutzern die Netzlast stark ansteigt.

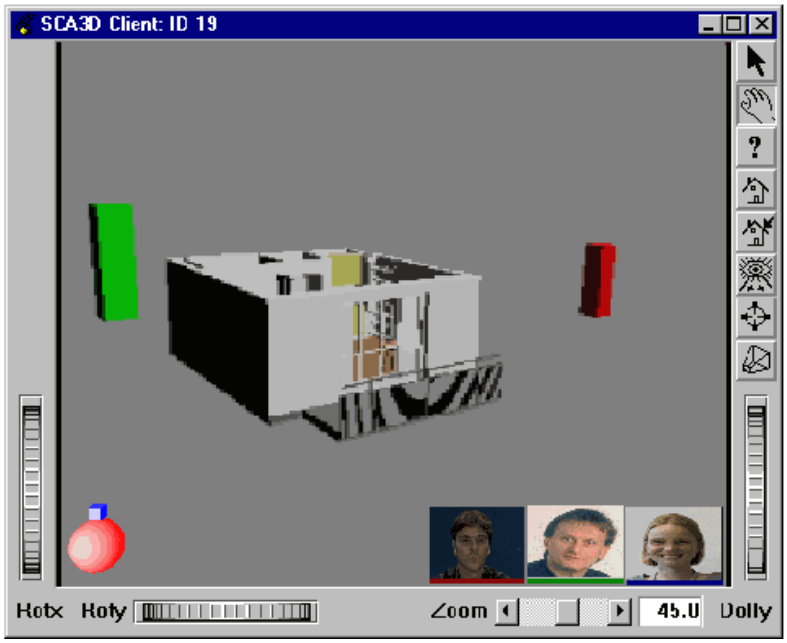

Abbildung 9.3: Avatare (grüner und roter Quader) von zwei Benutzern gesehen von einem dritten Teilnehmer.

In der SCA3D-Architektur werden Avatare in die Server-Szene integriert und die Synchronisation erfolgt über die Kameraparameter, die für die Navigation zum Server gesendet werden. Auch bei einer ansteigender Anzahl von Benutzern wird die Netzlast auf diese Weise nicht zusätzlich zunehmen. Das Avatar-Objekt für einen Benutzer wird dazu bei der Anmeldung in die Server-Szene als Gruppe eingefügt, die aus einem Separatorknoten, einem Transform-Knoten, einem Materialknoten und dem geometrischen Avatar-Objekt besteht. Durch Synchronisation der jeweiligen Kameraparameter mit dem Transform-Knoten wird das Avatar-Objekt stets entlang der Navigationroute des Benutzers geführt und entsprechend ausgerichtet. Über den Materialknoten wird die Farbe, die dem Benutzer zugeordnet ist, auf das Avatar-Objekt übertragen. In Abbildung 9.3 ist die graphische Oberfläche einer Client-Anwendung gezeigt, deren Benutzer mit zwei anderen Teilnehmern eine kooperative Konferenz durchführt. Die Anwesenheit der Benutzer ist dort durch Avatare, hier einfache Quaderobjekte, kenntlich gemacht. Die Avatare bewegen sich in der Server-Szene geführt durch die Navigation der Benutzer. Die Farbe der Avatar-Objekte entspricht dabei der farblichen Codierung, die für die einzelnen Benutzer vergeben wurde. Der Avatar des Benutzers selbst ist in diesem Fall über der Kameraposition und deshalb für ihn nicht zu sehen. In der SCA3D-Architektur wurden lediglich einfache Versuche mit diesen Avatar-Objekten gemacht, deren Ergebnisse in Abbildung 9.3 gezeigt sind.

9.3 Zugriffstransparenz und Synchronisation von 3D Objekten

In diesem Abschnitt soll es um Methoden zur Aufrechterhaltung eines einheitlichen Zustandes der Dokumentobjekte gehen, die zwischen Server und Client-Anwendungen verteilt sind. Bei der Bearbeitung eines Dokumentes durch die Benutzer wird fortlaufend der Zustand der Objekte verändert. Dies geschieht z.B. durch Manipulation der Objektparameter oder durch Transformation bezüglich eines der sechs Bewegungsfreiheitsgrade. Auch das Verhalten eines Objektes kann sich ändern, wenn z.B. die Zugriffsrechte wechseln. Schließlich kann das Dokumentobjekt am Server sich durch Interaktion so ändern, daß die graphische Repräsentationsform in der Client-Szene nicht mehr durch entsprechende Manipulation angepaßt werden kann. Bei parametrischen Objekten am Server und 3D Netzrepräsentationen am Client kann dieses Problem auftreten, da Änderungen der Parameter nicht immer direkt in Operationen am 3D Netz umgesetzt werden können. In diesem Fall kann nur durch vollständiges Ersetzen der Objektrepräsentation das Dokumentobjekt wieder ausreichend genau repräsentiert werden. Alle diese Änderungen an Zustand, Verhalten und graphischer Repräsentationsform von Objekten müssen in der verteilten Umgebung synchronisiert werden. Auf diese Weise wird durch das System auch eine Zugriffstransparenz realisiert, d.h. für den Benutzer macht es im besten Fall keinen Unterschied, ob er auf lokale oder entfernte Objekte zugreift.

9.3.1 Synchrone Interaktion mit 3D Objekten durch Transform- und Manipulator-Knoten

Durch Verwendung von Transform-Knoten in szenengraph-basierten Visualisierungsanwendungen können sehr viele Transformationen auf beliebige 3D Objekte angewendet werden. Der *SoTransform*-Knoten der OpenInventor-Bibliothek erlaubt die Verschiebung, Rotation und Skalierung von 3D Objekten. Die Skalierung ist durch Angabe des Zentrums, der Orientierung und eines Faktorvektors sehr flexibel einsetzbar. Um nun das server-seitige 3D Objekt entsprechend den Manipulationsaktionen des Benutzers zu synchronisieren, wird innerhalb des Interface-Objektes am Server

ein Transform-Knoten in die Kopie des aktivierten Gruppenknotens integriert, wie es in Abbildung 9.4 gezeigt ist.

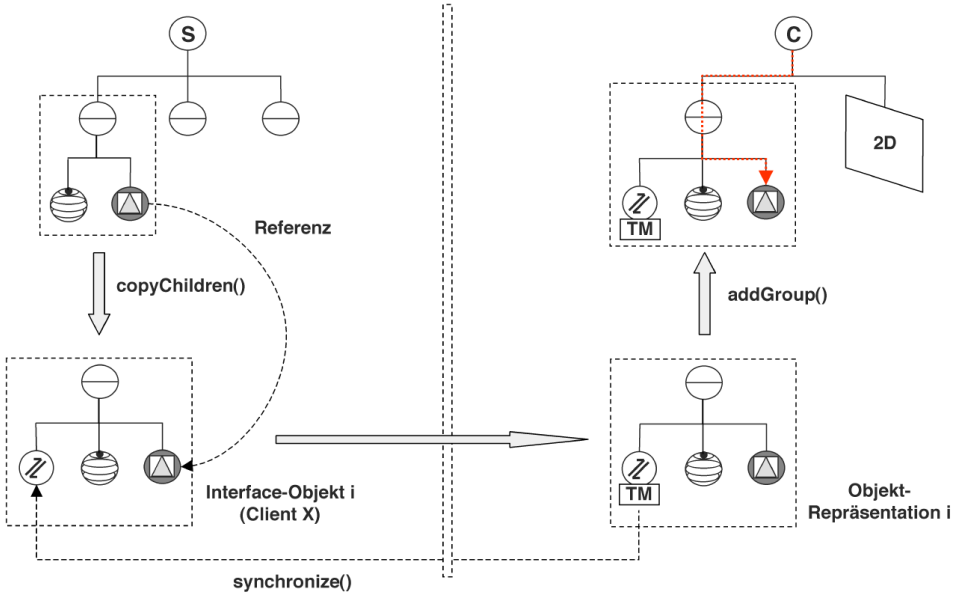

Abbildung 9.4: Synchronisierung von Objekten mittels der OIV-Klassen *SoTransform* und *SoTransformerManip.*

Beim Anlegen des Interface-Objektes wird eine Kopie der aktivierten Objektgruppe angelegt, indem alle Child-Knoten aus der Originalgruppe an einen neuen Gruppenknoten als Kopie angehängt werden. Wenn nun ein Transform-Knoten vor allen Child-Knoten eingefügt wird, werden alle Transformationen über die Objektreferenzen auf das Originaldokument übertragen. Hierdurch wird erreicht, daß das Originaldokument nicht mit zusätzlichen Knoten versehen werden muß und trotzdem der Zugriff auf die Objekte möglich wird. Nun wird die Objektrepräsentation, die durch das Interface-Objekt erzeugt wird, zur Client-Anwendung übertragen, wobei ein *SoTransformerManip*-Knoten vor den Child-Knoten eingefügt wird. Die Objektrepräsentation wird dann in die Client-Szene integriert und auf dem Benutzerbildschirm sichtbar. Der eingefügte Manipulator-Knoten bewirkt, daß eine Benutzer direkt durch das Bewegen von Ansetzpunkten des visuellen Manipulator-Objektes mit dem Eingabegerät das selektierte 3D Objekt verschieben, rotieren und skalieren kann.

Die Software-Architektur SCA3D sorgt nun dafür, daß der Manipulator-Knoten auf Client-Seite und der Transform-Knoten auf Server-Seite durch Austausch der Transformationsparameter synchronisiert werden. Dazu wird eine CORBA-Interface erzeugt, ein Implementierung entsprechender Methoden zum setzen der Feldwerte des Transform-Knotens auf dem Server abgelegt und eine Objektreferenz in der Client-Anwendung bereitgestellt. Alle Aktionen die der Benutzer nun mittels des Manipulator-Knotens auf die Objektrepräsentation anwendet, werden über RPCs direkt auf das Objekt im Originaldokument übertragen. Insbesondere hat dieses Vorgehen den Vorteil, daß auch unterschiedliche graphische Repräsentationsformen im Interaktions- bzw. Navigationsraum auf einheitliche Weise verändert werden können. Allerdings besteht der Nachteil, das keine Objektparameter direkt beeinflußt werden können, sondern Manipulation immer nur über den Umweg des

Transform-Knotens stattfinden kann. Wenn sich die Interaktion des Benutzers auf das Verschieben oder Rotieren von Objekten beschränkt, kann diese Technik in allen Fällen angewendet werden. Daher werden innerhalb der SCA3D-Architektur diese einfachen Aktionen auf die beschriebene Weise realisiert. Um einen flexibleren Zugriff auf alle Parameter zu ermöglichen, wird im nächsten Abschnitt ein allgemein anwendbares Verfahren vorgestellt.

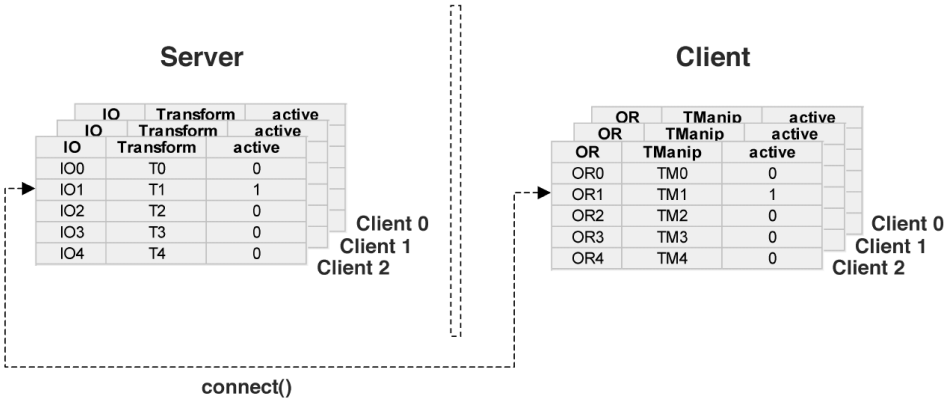

Abbildung 9.5: Verwaltung der Referenzen auf die Transformer auf Server- und Client-Seite.

Um den Zugriff auf ein Dokument für mehrere Benutzer gleichzeitig zu ermöglichen, werden auf Server- und Client-Seite Listen der Interface-Objekte (IO) bzw. Objektrepräsentationen (OR) aller Benutzer verwaltet. Jedes Interface-Objekte enthält eine Referenz auf den Transform-Knoten (T) des aktivierten Objektes, genauso wie jedes OR-Objekt eine Referenz auf den entsprechenden Manipulator (TM) bereitstellt. Abbildung 9.5 veranschaulicht diese Listenverwaltung. Wenn ein Benutzer eine Objektrepräsentationen selektiert, wird der dazugehörige Manipulator als aktiv gekennzeichnet und über das CORBA-Interface mit dem entsprechenden Transform-Knoten im Interface-Objekt verbunden. Dazu wird eine Namensschema verwendet, das eine eindeutige Zuordnung von Objektrepräsentation und Interface-Objekt erlaubt. Dieses Namensschema wurde bereist in Abschnitt 6.3.2 im Zusammenhang mit dem Document-Request-Broker beschrieben, der hier zum Einsatz kommt. Jedem Benutzer wird auf diese Weise jeweils eine aktive Verbindung zwischen Client- und Server-Szene bereitgestellt, die dynamisch wechselt, sobald ein anderes 3D Objekt bearbeitet werden soll. Der Benutzer muß dafür lediglich durch Anklicken das neue Objekt vollständig aktivieren.

9.3.2 Zugriff auf Objektparameter über dynamisch erzeugte Aufrufschnittstellen

Bei der verteilten Bearbeitung von 3D Dokumenten soll dem Benutzer auch die Möglichkeit gegeben werden, allgemeine Parameter eines 3D Objektes zu beeinflussen. Diese Parameter können z.B. die Farbe oder die Reflektionseigenschaften bestimmen, wichtiger sind aber noch Parameter, die die geometrische Form von parametrischen Objekten festlegen. In dem objekt-orientierten Dokumentmodell eines Szenengraphen werden diese Parameter als Felder in verschiedenen Knoten abgelegt. Oberflächeneigenschaften sind als Felder von Materialknoten (*SoMaterial*) abgelegt, Geometrieparameter (z.B. Radius, Höhe, Tiefe, Breite, etc.) sind als Felder von Formknoten (*SoShape*) angegeben und Kontrollelemente (z.B. Kontrollpunkte einer NURBS Fläche) in Feldern von Koordinatenknoten (*SoCoordinate3*) definiert. Durch Anordnung dieser Knoten innerhalb einer Szenen-

graphgruppe wird das 3D Objekt mit seinen speziellen Eigenschaften festgelegt und wird dementsprechend auf dem Bildschirm dargestellt.

Wenn ein Benutzer auf ein entferntes 3D Dokument auf dem SCA3D-Server zugreift und Objekte aktiviert, hat er zunächst keine direkte Information über die zugreifbaren Parameter und deren Auswirkung auf das 3D Objekt. Bei allen Interaktionen, die über das Verschieben, Rotieren und einfaches Skalieren hinausgehen, benötigt der Benutzer daher Angaben über Methoden mit denen er diese Parameter beeinflussen kann. Insbesondere muß die korrekte Form (Syntax) und die Bedeutung eines Methodenaufrufs (Semantik) in einer verständlichen und durch die Client-Anwendung umsetzbaren Form mitgeteilt werden. Im einfachen Fall eines Quaderobjektes sollen dem Benutzer also durch seine Client-Anwendung drei Bedienelemente zur Verfügung gestellt werden, mit denen er die drei Kantenlängen (Höhe, Tiefe, Breite) des server-seitigen Objektes kontrollieren kann. Dies kann z.B. durch eine Eingabefenster für Zahlenwerte geschehen, wobei für jedes Eingabefeld der Methodenname angegeben wird. Um vernünftige Werte für einen Parameter eingeben zu können, sollte der Benutzer auch ein Vorstellung von der Bedeutung des Parameters haben. Wenn z.B. der Radius einer Kugel verändert werden kann, sollte der angegebene Methodenname es dem Benutzer ermöglichen, die Bedeutung seiner Interaktion zu erkennen. Die direkte Lösung für dieses Problem ergibt sich, indem die angezeigten Methodennamen der Schnittstelle zum Dokument durch eine Kombination aus Aktion, Feldname und Knotentyp gebildet werden. Also ergibt sich z.B. im Fall des Kugelradius ein Methodenname *setRadiusSoSphere*. Wenn es mehrere Parameter mit ähnlicher Bedeutung gibt, z.B. bei einer NURBS-Fläche mit mehreren Kontrollpunkten werden die Feldnamen mit einer laufenden Zahl versehen, also z.B. *setPoint1SoCoordinate3,* wobei in diesem Fall die Feldnamen der Kontrollpunkte durchnumeriert werden.

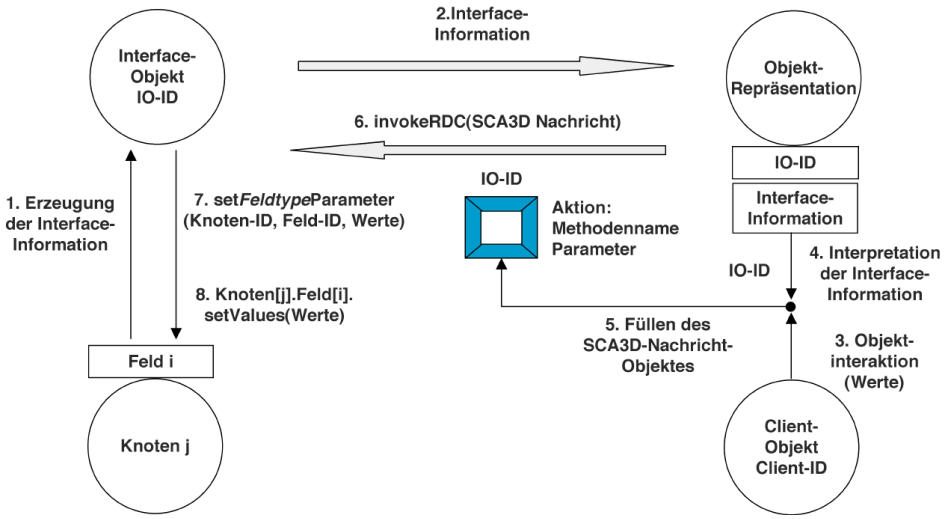

Abbildung 9.6: Aufruf entfernter Methoden über dynamisch erzeugte Schnittstellen.

Intern wird die Syntax der Methodenaufrufe in einer Form angegeben, die der Sprache IDL des CORBA-Standards entspricht. Dabei wird nicht das obige Namensschema benutzt, das zur Verdeutlichung der Methodenbedeutung für den Benutzer dient, sondern es wird die Methodensyntax der im Interface-Objekt vorhandenen Methode angegeben. Im Interface-Objekt sind für alle vorkommenden Feldtypen Methoden zu deren Manipulation implementiert, sodaß z.B. zum Ändern eines

Kugelradius die Methode *setFloatParameter(in NodeIndex, in FieldIndex, in float r)* aufgerufen werden kann. Eine Schnittstellendefinition in Form einer Textkette würden dann folgendermaßen aussehen:

"N-NodeIndex: NodeType; F-FieldIndex: FieldType; setFieldnameNodetype; Return-Parameter-List setFieldtypeParameter(Parameter-List); N-NodeIndex:..."

Diese Angaben über die Schnittstelle eines aktivierten Objektes werden bei der Erzeugung der Objektrepräsentation als Zeichenkette in einen Label-Knoten abgelegt, der in die übertragenen Objektdaten integriert wird. Die Client-Anwendung liest die Schnittstellendefinition aus dem Label-Knoten und bietet dem Benutzer dementsprechend Eingabeelemente für die einzelnen Parameter an.

Dynamische Erzeugung der Schnittstelleninformation

Die Schnittstelleninformation wird bei der Erzeugung des Interface-Objektes am Server aus den Informationen des Szenengraphen erstellt. Wenn ein 3D Objekt aktiviert wird und dabei eine Kopie der dazugehörigen Objektgruppe im Interface-Objekt abgelegt wird, kann die benötigte Information über die einzelnen Child-Knoten und deren Felder dabei abgerufen werden. Dafür sind wie oben beschrieben, der Knotenindex bzw. -typ, der Feldindex, -name bzw. -typ und die Aufrufdefinition der Server-seitigen Methode nötig. Damit eine Transparenz bezüglich der Implementierung möglich wird, sollte die Aufrufdefinition unabhängig von speziellen lokalen Datentypen sein. OpenInventor verwendet z.B. den Typ $SoSFFloat$ zur Angabe eines Float-Parameters. Eine Client-Anwendung sollte aber über die angegebene Schnittstelle einfache Datentypen verwenden können. Dies wird beim CORBA-Standard durch die Sprache IDL erreicht, die einen Satz an Basisdatentypen festlegt. Beim übersetzen der IDL-Definition in die jeweilige Programmiersprache werden die Datentypen dann angepaßt. Damit allgemeine Datentypen (z.B. $Float$) von entfernten Anwendungen gesetzt werden können, werden im Document-Request-Broker innerhalb des Interface-Objektes Methoden dafür direkt implementiert (z.B. $setFloatParameter(Field, value)$). Innerhalb der Methode wird dann die implementierungsabhängige Methode zum Setzen des Parameters aufgerufen (z.B. Field.setValue(value)). Die Zuordnung von einem Feld des speziellen Datentypes zu der allgemeinen Methode wird bei der Erzeugung der Schnittstelleninformation vorgenommen (z.B. $SoSFFloat \rightarrow Float \rightarrow setFloatParameter$). Dazu wird eine Liste verwendet, die jeden speziellen Datentyp mit einem Basisdatentyp und der dazugehörigen im Interface-Objekt implementierten Methode verknüpft. Die Syntax der Methode, also Eingabeparameter, Rückgabeparameter und der Methodenname, wird dann in der Schnittstellendefinition abgelegt und der Client-Anwendung mitgeteilt. Durch Interpretation dieser Information kann die Client-Anwendung dann die entsprechenden Parameter füllen und mit Hilfe einer SCA3D-Nachricht einen Remote-Document-Call absetzen.

Für ein beliebiges Dokumentobjekt kann auf diese Weise die Schnittstelleninformation dynamisch erzeugt werden, wobei die Information dann mit der Objektrepräsentation übertragen wird. Im CORBA-Standard wird für dynamische Aufrufe von Remote-Methoden zur Laufzeit, bei denen keine Stub-Objekte im voraus erzeugt werden müssen, das sogenannte *Dynamic Invocation Interface* (DII) verwendet. Dazu schreibt die Server-Anwendung die Interface-Information in das sogenannte Interface-Repository, das Server und Client über den ORB zugänglich ist. In einer CORBA-Anwendung geschieht der Aufruf einer beliebigen Operation mittel des DII in folgenden Schritten:

- Beschaffung der Objektreferenz für das aufzurufende Objekt (z.B. Naming-Service).
- Abfrage der Schnittstelleninformation aus dem Interface-Repository

- Erzeugen eines Request-Pseudo-Objekts mindestens mit Angabe des aufzurufenden Objektes, des Methodennamens und der Parameterliste.
- Übergabe des Request-Pseudo-Objekts an den ORB, Invoke-Ausführung.
- Optionales Warten auf das Ergebnis der Operation.
- Vernichten oder Wiederverwenden des Request-Pseudo-Objekts.

Um eine ähnliche Flexibilität beim Aufruf von entfernten Methoden der Interface-Objekte innerhalb der SCA3D-Architektur zu erreichen, werden die oben beschrieben Techniken zur dynamischen Schnittstellenerzeugung verwendet. Ein kurzer Vergleich der hier verwendeten Mechanismen mit den einzelnen Schritten des DII soll die Ähnlichkeit des Vorgehens unterstreichen.

```
...
typedef  sequence <long> longs;
typedef  sequence <float> floats;
typedef  sequence <string> strings;

struct Param {                              // RDC Method parameters
        longs   intInput;
        floats  floatInput;
        strings stringInput;
  };

struct SCAMessage {                         // Four-sided SCA3D Message
        long   Id;                          // User Identifier
        string IOID;                        // Object to be interacted
        string RDCMethodName;               // RDC Method and Parameters
        Param  RDCParams;
        float  quality;                     // Client's performance
        float  level;
        string rights;                      // Client's rights situation
  };

interface SceneControl {
        Param   invokeRDC(in SCAMessage scam);
        ...
};
...
```

Abbildung 9.7: Ausschnitt aus dem IDL-Interface zum Aufruf von Remote-Document-Calls.

Die Beschaffung der Objektreferenz erfolgt in SCA3D bei der Selektion einer Objektrepräsentation, deren Root-Knoten mit einem eindeutigen Namen versehen ist, der Auskunft über das entsprechende server-seitige Interface-Objekt gibt. Die Abfrage der Schnittstelleninformation passiert dann nicht bei jedem Methodenaufruf neu, sondern die Schnittstelleninformation wird aus dem Label-Knoten gelesen, in den diese Information bei der Erzeugung des Interface-Objektes eingetragen wurde. Mit Hilfe der Schnittstelleninformation wird dann ein SCA3D-Message-Objekt (vgl. Abschnitt 6.2) erzeugt, das ähnlich dem Request-Pseudo-Objekts in CORBA die wichtigsten Informationen für den entfernten Aufruf der gewünschten Methode enthält. Das SCA3D-Message-Objekt wird als Parameter an einen allgemeinen *invoke*-Aufruf des Document-Request-Brokers übergeben. Auf Server-Seite wird die enthaltene Information zum Aufruf der Methode des Interface-Objektes verwendet, die der Benutzeraktion entspricht. Die Ergebnisse der Operation werden in Form eines Parameter-Objektes als Rückgabe des *invoke*-Aufrufs an die Client-Anwendung übergeben. Wenn ein SCA3D-Message-Objekt nicht mehr verwendet wird, wird es durch die Java-Virtual-Machine automatisch vernichtet. Für einen erneuten Aufruf wird ein neues SCA3D-Message-Objekt erzeugt.

In Abbildung 9.7 ist ein Ausschnitt aus dem IDL-Interface gezeigt, über das mit Hilfe der CORBA-Implementation die beschriebenen Remote-Document-Calls aufgerufen werden können. Die Struktur der SCA3D-Nachricht mit ihren Komponenten ist zu erkennen.

Dieser Vergleich zeigt, daß die Vorgehensweise innerhalb der SCA3D-Architektur derjenigen des Dynamic-Invocation-Interace, wie es in CORBA definiert ist, sehr ähnelt. Der Document-Request-Broker der SCA3D-Architektur erweitert dabei das DII-Konzept für den Einsatz beim verteilten Arbeiten mit 3D Dokumenten.

Bereitstellung von GUI-Komponenten zur Parametermanipulation

Um die Flexibilität des Document-Request-Broker-Ansatzes dem Benutzer zur Verfügung zu stellen, sollten intuitive Eingabemöglichkeiten für die Manipulation von Parametern über bestimmte Methoden realisiert werden. Neben den oben erwähnten Texteingabfeldern mit Angabe der Methodennamen, die die Bedeutung der Methode verdeutlichen, kann auch über 3D Objekte selbst die Parametermanipulation stattfinden. Wenn der Radius eines Objektes geändert werden kann, könnte mittles eines 3D Schiebereglerelementes mit einem Kugelobjekt als Positionsmarke, eine intuitive Eingabe erfolgen. Bei Änderung der Reglerposition würde sich dann das Kugelobjekt vergrößern bzw. verkleinern und damit verdeutlichen, daß es sich bei dem manipulierten Parameter um einen Radius handelt. Für Kugel-, Konus-, oder Ellipsen-Objekte ist diese Methode sinnvoll. Für die Änderung eines Kontrollpunktes ist die Verwendung eines Translations-Manipulators, der eine Verschiebung in die drei Grundrichtungen erlaubt, eine gute Lösung. Prinzipiell können die von OpenInventor angebotenen Manipulator-Knoten, von denen es eine große Vielfalt gibt, für diese Zwecke erfolgreich eingesetzt werden.

9.3.3 Anpassung der Objektrepräsentationen in den Interaktionsräumen

Damit die Objekte im Navigations- und Interaktionsraum synchronisiert bleiben, müssen die Aktionen des Benutzers, die durch Parameteränderungen hervorgerufen werden, auch auf die Objektrepräsentation übertragen werden. Wenn das client-seitige Objekt die gleiche graphische Repräsentationsform wir das Dokumentobjekt besitzt, dann können die Parameter mit Hilfe der Schnittstelleninformation lokal direkt angepaßt werden. Dazu müssen lediglich die Methoden zum Setzen der verschiedenen Datentypen im OR-Objekt implementiert sein, also z.B. *setVec3FloatParameter(in NodeIndex, in FieldIndex, in int index, in float x, in float y, in float z)* zum Setzen des Kontrollpunktes mit laufender Kennzahl *index* einer NURBS-Fläche.

Für den Fall, daß die graphische Repräsentationsform des Client-Objektes eine andere als die des Dokumentobjektes ist, muß die Anpassung im Interaktionsraum auf andere Weise erfolgen. Wenn z.B. am Server ein parametrisches Objekt über seine Parameter manipuliert wird und am Client ein 3D Netz als Objektrepräsentation verwendet wird, kann die Geometrieoperation nicht einfach übertragen werden. Als alternative Lösung kann auf den Manipulator-Knoten zurückgegriffen werden, der zur Verschiebung und Rotation in die Objektgruppe eingefügt wurde. Indem am Server vor und nach der Parameteränderung die Bounding-Box ermittelt wird, kann ein Skalierungsfaktor für jede der drei Kanten der Bounding-Box berechnet werden. Wenn a, b, c die Kantenlängen vor der Operation waren und a', b', c' die Kantenlängen danach, dann ergeben sich die Skalierungsfaktoren als $s_a = \frac{a'}{a}$, $s_b = \frac{b'}{b}$ und $s_c = \frac{c'}{c}$. Diese Skalierungsfaktoren werden als Rückgabeparameter der Operation zum Client übertragen und können dann direkt als Eingabeparameter einer Skalierungsoperation des Manipulator-Knoten angewendet werden. Für relativ einfache Objekte kann diese Anpassung über mehrere Operationsschritte ausreichend sein, um die Objektre-

präsentation an das Originaldokument anzupassen. Sobald die Abweichungen aber zu groß werden, muß der Benutzer die Möglichkeit haben, die Objektrepräsentation aktiv aktualisieren zu können. Dies geschieht durch Auslösen einer *Update*-Operation, die das Objekt deaktiviert und mit dem vorher gespeicherten Level-of-Information neu aktiviert.

Bei der Kooperation von mehreren Benutzern besteht die Möglichkeit, daß ein Dokumentobjekt von zwei oder mehr Teilnehmern aktiviert wird. Diese arbeiten dann lokal mit jeweils einer Objektrepräsentation des gleichen Dokumentobjektes. Damit Aktionen von einem Benutzer, die auf das server-seitige 3D Objekt übertragen werden, sich auch auf die Client-Objekte der anderen Benutzer auswirken, müssen die Operationen jeweils auch an die anderen Client-Anwendungen übergeben werden und dort auf die Objektrepräsentation angewendet werden. Da sich die graphischen Repräsentationsformen der verschiedenen Benutzer aber unterscheiden können (parametrische Objekte, 3D Netze, Bounding-Box-Information), können nicht alle Operationen auf die gleiche Weise synchronisiert werden. Um trotzdem ein geeignetes Verfahren zur Synchronisation zwischen gleichzeitig aktivierten Objekten zu integrieren, wird der oben beschriebene Mechanismus zur Anpassung der client-seitigen Objekte auf mehrere Client-Anwendungen erweitert. Dabei wird aufgrund der verschiedenen Objektrepräsentationen der Benutzer immer die Skalierungsmethode über den Manipulator-Knoten verwendet. Dieses Verfahren bietet sich auch an, weil dazu lediglich ein statisches Interface zu jedem Client aufgebaut werden muß, über das die Synchronisation des Manipulator-Knotens der aktiven Objektrepräsentation ablaufen kann. Diese Schnittstelle wird für jeden Benutzer bei der Anmeldung aufgebaut und funktioniert wie die schon beschriebene CORBA-Schnittstelle zur Synchronisation der Client-Kameras im Mehrbenutzerbetrieb. Jeder Benutzer kann seine Objektrepräsentation außerdem aktualisieren, sobald die Übereinstimmung mit dem Server-Objekt, die über im Navigationsraum beobachtet wird, nicht mehr ausreicht.

9.4 Ergebnisse: Kooperative Arbeitsprozesse mit dem Level-of-Information-Ansatzes

Nachdem die Anforderungen und Konzepte der SCA3D-Architektur in Mehrbenutzeranwendungen diskutiert wurden, sollen nun praktische Ergebnisse vorgestellt werden, die in der Testumgebung gewonnen wurden. Zuerst wird die Prototypanwendung von SCA3D daraufhin untersucht, inwieweit sie die Anforderungen an kooperative Systeme erfüllt. Dann werden anhand des Beispiels eines kooperativen Arbeitsprozesses innerhalb der SCA3D-Architektur praktische Ergebnisse vorgestellt.

Auswertung der SCA3D-Architektur bezüglich der Anforderungen

Am Beginn des Kapitels wurde für ein kooperatives System folgende Basisfunktionalität gefordert: Hinzufügen und Entfernen von 3D Objekten, Synchronisation von veränderten Objektzuständen bei allen Teilnehmern und Möglichkeiten zur audio/visuellen Kommunikation der Benutzer. Als fortgeschrittene Anforderungen wurden genannt: Anpassung an die Randbedingungen der heterogenen Umgebung und Techniken zur Gewährleistung der Sicherheit von 3D Daten und Rechnern. Der Prototyp der SCA3D-Software-Architektur erfüllt diese Anforderungen, wobei die angewendeten Lösungen integraler Bestandteil der Architektur sind. Lediglich für die audio/visuelle Kommunikation zwischen den Benutzern werden externe Techniken benutzt, die parallel zu den Prozessen der SCA3D-Architektur ablaufen.

Die SCA3D-Architektur wurde im Hinblick auf das Arbeiten mit 3D Dokumenten entworfen, dadurch sind Möglichkeiten zum Hinzufügen und Entfernen von 3D Dokumenten in bestehende

Dokumente integraler Bestandteil des Prototypen. Um die Arbeit in offenen Informationsräumen flexibel zu gestalten, wurde der Ansatz eines Document-Request-Brokers eingeführt und umgesetzt. Dieser Ansatz zusammen mit dem kooperativen Modell der SCA3D-Architektur sorgt dafür, daß alle Dokumentobjekte und Avatare innerhalb des verteilten Systems zu jeder Zeit synchronisiert werden. Die Forderung nach einer Anpassung an die Randbedingungen der heterogenen Umgebung wird durch den Level-of-Information-Ansatz realisiert. Dazu werden die Objektrepräsentationen des Document-Request-Brokers mit variablem Informationsgehalt zu den Client-Anwendungen verteilt. Das Sicherheitsmanagement wird ebenfalls durch die Kontrolle des Informationsgehalts der 3D Objekte am Client umgesetzt, wobei eine Konvertierung der Dokumentobjekte am Server stattfindet. Die Sicherheit der beteiligten Rechner wird soweit möglich durch eine Authentifizierung beim Zugriff auf den SCA3D-Server kontrolliert.

Um den Benutzern auch die Möglichkeit der Kommunikation über Audio/Video-Verbindungen zu geben, wurden Tests mit den MBone-Tools für Audiokonferenzen und Videokonferenzen durchgeführt (siehe auch [Kum96]). Die MBone-Tools sind über den Web-Server des *University College London UCL* erhältlich und wurden von mehreren Instituten entwickelt, darunter das *University College London* und das *Lawrence Berkeley National Laboratory*. Die freiverfügbaren Werkzeuge, zu denen unter anderem eine Anwendung zur Sitzungverwaltung (*SDR: Session Directory*), ein Video-Tool (*VIC: Videoconferencing Tool*) und ein Audio-Tool (*RAT: Robust Audio Tool*) gehören, wurden auf allen Rechnern der Testumgebung installiert und eine Audio/Video-Konferenz zur Unterstützung des Arbeitsprozesses mit SCA3D durchgeführt. Kommunikation von Benutzern während der verteilten Arbeit mit 3D Dokumenten ist ein wichtiges Hilfmittel, das die wirkungsvolle Durchführung von kooperativen Konferenzen über Computernetze möglich macht.

Die SCA3D-Architektur erfüllt alle genannten Anforderungen und verfügt durch den kombinierten Ansatz von lokaler und Remote-Visualisierung über ein hohes Maß an Flexibilität beim kooperativen Arbeiten mit 3D Dokumenten. Der Begriff des 3D Dokumentes in verteilten Umgebungen steht dabei im Mittelpunkt und wurde bei der Entwicklung der Architektur als wichtigste Orientierung verwendet.

Beispiel eines kooperativen Arbeitsprozesses

Als praktisches Beispiel soll nun eine kooperative Konferenz zweier Benutzer anhand von Testergebnissen beschrieben werden, bei der ein 3D Dokument mit einem CAD-Kugellagermodell aus einzelnen Komponenten zusammengebaut werden soll. Als Ausgangssituation werden zwei 3D Dokumente bereitgestellt, die zum einen das Kugellagermodell und zum anderen eine Modell einer Achse enthalten. Das Kugellager ist anfangs noch nicht zusammengebaut, sondern liegt in Form von aneinandergereihter Einzelteile vor. Die beiden 3D Dokumente zusammen haben eine Größe von ungefähr zwei MByte und liegen auf dem Server im Filesystem bereit. In Abbildung 9.8 sind die Ausgangsdokumente gezeigt, die von zwei Benutzern mit Hilfe des Prototypen der SCA3D-Architektur zusammengefügt werden sollen. Die Einzelteile der 3D Modelle liegen im Original als parametrische Basisobjekte und 3D Netze vor, wobei OpenInventor-*IndexedFaceSet*-Knoten zur graphischen Repräsentation der 3D Netze verwendet wurden.

Die Aufgabe für zwei Benutzer, die sich mit ihren Rechnern an getrennten Orten befinden und über ein Netz mit dem SCA3D-Server verbunden sind, besteht darin, ein Gesamtdokument zu erzeugen, das das zusammengesetzte Kugellager mit der eingefügten Achse enthält und am Server abgelegt wird. Dabei dürfen die beiden Benutzer entsprechende ihrer Anbindung und Leistung Objektrepräsentationen der einzelnen Objekte in ihren jeweiligen Interaktionraum laden. Allerdings sollen die Benutzer versuchen, möglichst wenige Objekte in hoher Auflösung lokal zu verwenden,

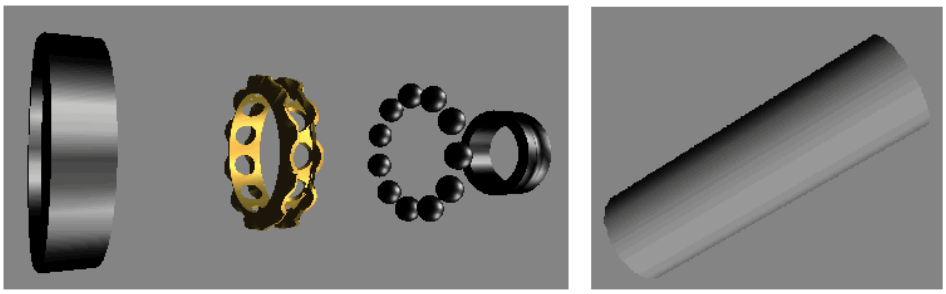

Abbildung 9.8: Ausgangsdokumente für die kooperative 3D Konferenz.

um die Last für das Netz und die lokalen Rechner möglichst gering zu halten.

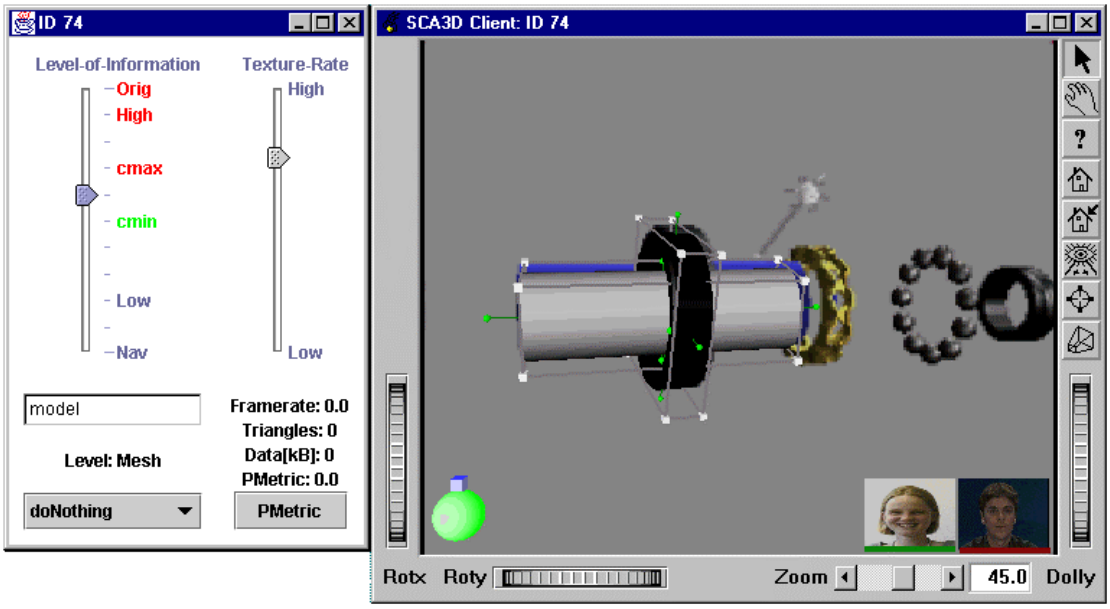

Abbildung 9.9: Screenshot einer SCA3D-Client-Anwendungen während des Arbeitsprozesses.

Die Aufgabe wird hier so erledigt, daß beide Benutzer anfangs das Kugellagerdokument durch Remote-Rendering visualisieren und ein Benutzer nach Absprache das Achsenobjekt in das Dokument einfügt. Dann bearbeitet ein Benutzer die Position der Achse und der äußeren Kugellagerschale, während der zweite Teilnehmer die beiden inneren Kugellagerschalen bearbeitet. Die Kugeln (3D Netze), die einzeln positioniert werden müßten, bleiben bei dem Arbeitsprozeß unverändert. Um die inneren Teile des Kuglelagers zusammenzubauen zu können, aktiviert Benutzer 2 diese beiden Objekte und verschiebt sie so, daß sie innerhalb der Gruppe der Kugeln eingepaßt sind. Benutzer 1 aktiviert inzwischen die äußere Schale und die Achse und bringt beide durch Verschieben der Achse in die richtige relative Position. Dann aktiviert er nachdem Benutzer eins seine Arbeit beendet hat, eine innere Kugellagerschale und positioniert Achse und äußere Schale

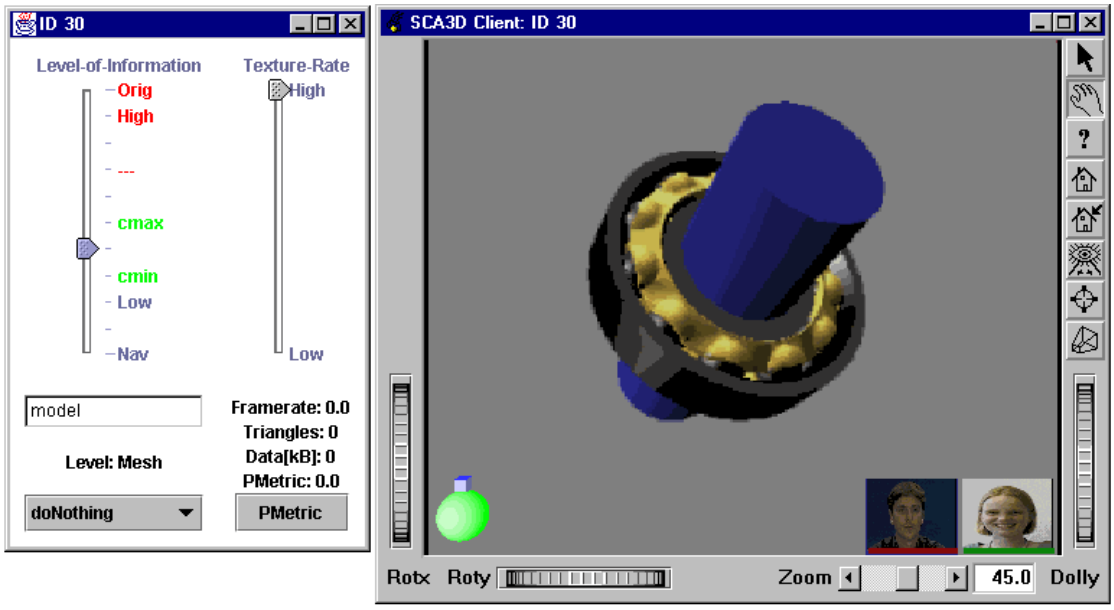

Abbildung 9.10: Screenshot der Client-Anwendungen des anderen Teilnehmers nach Fertigstellung des Dokumentes.

mit Hilfe der inneren Schale richtig. Am Ende des Arbeitsprozesses ist das Kugellager zusammengefügt und die Achse an der richtigen Stelle eingebaut, wobei beide Benutzer mit relativ wenig geometrischer Information in ihrer lokalen Szenen ausgekommen sind. Dadurch lassen sich die zu übertragende Datenmenge und die lokale Last möglichst gering halten und Teile, die nicht als Original übertragen werden dürfen, werden vor unkontrollierter Weiterverbreitung geschützt. Als letzten Arbeitsschritt deaktivieren die Benutzer alle Objekte und speichern das fertige Dokument, unter neuem Namen auf dem Server ab. In den Abbildungen 9.9 und 9.10 sind Screenshots der SCA3D-Client-Anwendungen gezeigt, die während des beschriebene Arbeitsprozesses von beiden Benutzern aufgezeichnet wurden. Die gesperrten Bereiche des LoI-Reglers sind dabei durch ein rote Einfärbung der Beschriftung gekennzeichnet. Die empfohlenen Positionen des Reglers sind durch die Grenzen c_{min} und c_{min} angegeben und durch eine grüne Einfärbung kenntlich gemacht. Falls die Grenzen im gesperrten Bereich liegen werden sie dementsprechend rot eingefärbt (siehe Abbildung 9.9). Abbildung 9.10 zeigt das fertige 3D Dokument nach Bearbeitung durch die beiden Teilnehmer der Konferenz.

9.5 Zusammenfassung

In diesem Kapitel wurde als zweiter wichtiger Aspekt der SCA3D-Architektur das kooperative Arbeiten mit 3D Dokumenten untersucht. Dazu wurden die Anforderungen an verteilte Systeme für kooperative Konferenzen beschrieben, bei denen Mehrbenutzervisualisierung von 3D Szenen verwendet wird. Es wurden Basisanforderungen, wie *Cut-and-Paste* von Objekten, Objektsynchronisation und Benutzerkommunikation, und fortgeschrittene Anforderungen, wie Adaptivität des Verhaltens und Sicherheitsmanagement, definiert. Dann wurden kooperative Szenarien innerhalb eines

SCA3D-Systems diskutiert und der Nutzen von Navigationsraum und Interaktionsräumen für das kooperative Arbeiten ausgewertet. Möglichkeiten zur Benutzerführung durch Informationsvisualisierung, die im SCA3D-Prototypen umgesetzt sind, wurden ebenfalls vorgestellt. Im folgenden wurden die Ansätze zur Realisierung der Zugriffstransparenz und zur Synchronisation der Objekte in Navigations- und Intaktionsräumen im einzelnen untersucht. Dabei wurde die Realisierung von statischen und dynamischen Schnittstellen für den Zugriff auf server-seitige 3D Dokumente mit Hilfe des Document-Request-Brokers nocheinmal genauer beleuchtet. Abschließend wurden am Beispiel einer kooperativen Konferenz mit 3D Dokumenten Ergebnisse vorgestellt, die mit Hilfe des SCA3D-Prototypen gewonnen wurden. Im nun folgenden Abschlußkapitel wird eine Zusammenfassung der Arbeit und ein Ausblick auf weitere Forschungsarbeiten im Zusammenhang mit der SCA3D-Architektur gegeben.

Kapitel 10

Zusammenfassung und Ausblick

Als Abschluß werden nun die Vorgehensweise bei der Konzeption, Umsetzung und Implementierung der Software-Architektur noch einmal zusammengefaßt und die erreichten Ergebnisse bei der Bearbeitung des Themas mit den geplanten Zielen verglichen. Die neuen Aspekte im Hinblick auf den Stand der Forschung auf dem Gebiet verteilter Visualisierungsarchitekturen sollen noch einmal herausgestellt werden. Weiterführende Forschungsrichtungen im Hinblick auf kombinierte Visualisierungsansätze für 3D Dokumente in verteilten Umgebungen werden als Ausblick angesprochen. Darüber hinaus soll die Erweiterbarkeit des Konzeptes auf heterogene Dokumente, die neben 3D Szenen auch andere Inhaltstypen wie Audio- oder Videodaten enthalten, angesprochen werden. Die Verwendbarkeit von internaltionalen Standards zur Realisierung des beschriebenen Konzeptes wird ebenfalls kurz diskutiert.

10.1 Erreichte Ziele und Schlußfolgerungen

Das Arbeitsthema zielte auf den Entwurf und die Umsetzung einer neuartigen, verteilten Software-Architektur ab, die heutigen Anforderungen entsprechend integrierte Möglichkeiten zum Management von Komplexität und der Sicherheit von 3D Szenen bietet. Dabei sollte eine dynamische Kombination von lokaler und Remote-Visualisierung von 3D Szenen erlauben, einen optimalen Ausgleich der Verzögerungszeit beim Remote-Rendering durch eine hohe interaktive Bildwiederholrate für selektierte Objekte zu erreichen, die lokal visualisiert werden. Dadurch sollte es möglich werden, unterstützt durch das visuelles Feedback von einem Dokument- und Visualisierungs-Server mit geschützten und komplexen 3D Szenen in verteilten Umgebungen arbeiten zu können. Damit die Randbedingungen einer heterogenen Umgebung bezüglich Bandbreite und lokaler Leistung einzelner Client-Anwendungen berücksichtigt werden, sollte der Informationsgehalt auf Client-Seite regulierbar sein, der zum Arbeiten mit 3D Szenen benötigt wird. Dies wurde als Level-of-Information-Ansatz bezeichnet. Weiterhin sollte eine Transparenz des Zugriffs auf verteilte 3D Szenen realisiert werden, wofür ein objekt-orientiertes Dokumentmodell für 3D Szenen erfolgversprechend einsetzbar ist. Mit Hilfe der so gestalteten Software-Architektur sollten Verfahren realisiert werden, die eine adaptives Verhalten bezüglich der zu übertragenden und lokal zu berechnenden Datenmenge ermöglichen würden. Außerdem sollte ein Konzept zum Sicherheitsmanagement für 3D Szenen in der Architektur umgesetzt werden. Schließlich gehörte zu den Anforderungen an ein solches System auch die Möglichkeit, daß mehrere Benutzer synchronisiert mit 3D Szenen in verteilten Umgebungen arbeiten können. Zusammengefaßt war es also das Ziel, eine verteilte, kooperative Visualisierungsarchitektur für 3D Szenen zu entwerfen, die besonders an den Aspekten

Komplexität und Sicherheit von 3D Szenen ausgerichtet ist.

Mit diesen Anforderungen wurde daraufhin das Konzept der Skalierbaren Szenen erarbeitet und die Software-Architektur SCA3D entworfen. Ein Prototyp für diese Architektur wurde entwickelt und in einer Testumgebung ausgewertet. Die erreichten Ziele entsprechen den gestellten Anforderungen, was durch entsprechende Tests und Meßergebnisse verdeutlicht wurde. Für die Bearbeitung von 3D Szenen wurde der zentrale Begriff des 3D Dokumentes verwendet, das durch seine Eigenschaften besonders gut in den Aufbau der beschriebenen Software-Architektur paßt. Die technische Umsetzung des geforderten Systems geschah durch die Entwicklung eines Document-Request-Broker-Ansatzes für verteilte 3D Dokumente, die sich auf ein internes objektorientiertes Dokumentmodell, den Szenengraphen, abbilden lassen. Durch diesen Ansatz, der dem Konzept einer objekt-orientierten Middleware, wie etwa dem CORBA-Standard, ähnlich ist, läßt sich die geforderte Zugriffstransparenz für verteilte 3D Szenen realisieren. Weiterhin wird durch die Verwendung verschiedener graphischer Repräsentationsformen mit variablem Imformationsgehalt, die als Proxy-Objekte der entfernten Dokumentobjekte benutzt werden, der beschriebene Level-of-Information-Ansatz realisiert. Bei diesem Ansatz wird jeder Benutzer mit einem für seine Verhältnisse günstigen Informationgehalt versorgt und seine Client-Szene dementsprechend angepaßt. Dem Benutzer wird mit dem Level-of-Information-Schieberegler eine Möglichkeit zur Kontrolle des Informationsgehalt in seiner lokalen Szene gegeben. Durch Austausch von Objektrepräsentationen zwischen zwei synchronisierten Rendering-Pipelines auf dem Server- und dem Client-Rechner wird eine dynamischen Kombination von lokalem und Remote-Rendering erreicht.

Um ein adaptives Verhalten zu realisieren, wurde mit Hilfe einer Performanzmetrik die Steuerung des Level-of-Information-Reglers und der lokalen bzw. Remote-Frameraten ermöglicht. Diese Performanzmetrik wertet sowohl die Anbindungbandbreite, als auch die lokale Leistung der jeweiligen Client-Anwendung aus und reagiert auch auf veränderliche Leistung des Servers, da es sich bei der angewandten Technik um einen verteilten Regelkreis handelt. Für einen Benutzer bedeutet die Steuerung über die Performanzmetrik, daß ihm zu jeder Zeit ein Bereich optimaler Positionen des LoI-Reglers empfohlen wird und dementsprechend eine geeignete Position automatisch eingestellt wird. Allerdings kann ein Benutzer von dieser Empfehlung auch individuell abweichen. Im Gegensatz dazu sind bestimmte Positionen der LoI-Skala für den Benutzer mit einer vorgegebenen Rechtesituation gesperrt. Diese gesperrten Bereiche werden von dem integrierten Sicherheits-Management der SCA3D Software-Architektur festgelegt und verhindern eine Übertragung geschützter Objektrepräsentationen. Das Sicherheitskonzept arbeitet mit einer Authentifizierung der Benutzer, einem Rollenkonzept für verschiedene Benutzertypen und der Transformation von 3D Objekten in andere graphische Repräsentationsformen. Durch Verhandlung von Rechten zwischen Client und Server soll dem Benutzer die Möglichkeit gegeben werden, auch Originaldaten nach geänderter Rechtesituation zu erhalten. Diese Rechtekonzept läßt sich im Bereich digitaler Bibliotheken, bei dem Originale geschützt werden müssen, oder im Bereich von Produktschutz im Wirtschaftsumfeld erfolgreich einsetzen.

Schließlich wurde auch die geforderte Mehrbenutzertauglichkeit im Entwurf umgesetzt und im Prototyp der Architektur realisiert. Mehrere Benutzer können innerhalb der SCA3D-Architektur mit 3D Dokumenten arbeiten, wobei jeder Teilnehmer mit einer eigenen Konfiguration seiner Client-Szene interagiert. Dadurch läßt sich das System an die Heterogenität von verteilten Umgebungen anpassen und die Verzögerung zwischen ungleichen Client-Anwendungen minimieren. Ein kooperatives Modell zum Mehrbenutzerbetrieb ist Bestandteil der SCA3D-Architektur. Zur Auswertung der beschriebenen Konzepte und Techniken wurden Testläufe in der dafür aufgebauten Testumgebung durchgeführt und Meßergebnisse gewonnen. Diese Ergebnisse wurden vorgestellt und im Hinblick auf die Anforderungen diskutiert.

Aus dieser Zusammenfassung ergeben sich folgende neue Aspekte im Hinblick auf den aktuellen Stand der Forschung auf dem Gebiet der verteilten 3D Visualisierungsarchitekturen:

- *Verteilte 3D Dokumente als zentrale Idee*: Durch Einführung von 3D Dokumenten als zentrale Idee beim Entwurf und der Entwicklung einer verteilten Visualisierungsumgebung, werden die Eigenschaften von digitalen Dokumenten optimal unterstützt und die Funktionalität, die ein Benutzer beim Arbeiten mit Dokumenten erwartet, direkt in das unterstützende System integriert.

- *Dynamische Kombination von verteilten Rendering-Pipelines*: Durch den Austausch aller Objektrepräsentationen, die innerhalb einer Rendering-Pipeline verarbeitet werden, wird die dynamische Verteilung von Rendering-Prozessen ermöglicht. Die Rendering-Pipelines auf Server- und Client-Seite werden dabei über eine Middleware-Schnittstelle und das Streaming von Bilddaten synchronisiert.

- *Entwurf eines Document-Request-Brokers*: Die Übertragung eines objekt-orientierten Middleware-Konzeptes auf 3D Dokumente in verteilten Umgebungen, wird durch das objekt-orientierte Dokumentmodell eines Szenengraphen optimal unterstützt. Die Umsetzung eines Document-Request-Brokers für 3D Dokumente ist ein neues Konzept, das in dieser Arbeit vorgestellt wird.

- *Komplexitäts- und Sicherheitsmanagement durch den Level-of-Information-Ansatz*: Durch dynamische Verteilung von Objektrepräsentationen als Proxy-Objekte des entfernten 3D Dokumentes werden sowohl Möglichkeiten für den Umgang mit Komplexität, als auch zum Schutz von 3D Dokumenten vor unkontrollierter Verbreitung bzw. Vervielfältigung realisiert. Der beschriebene Level-of-Information-Ansatz macht diese Techniken in verteilten offenen Informationsräumen möglich.

10.2 Weiterführende Arbeiten

Das Konzept der Skalierbaren Szenen und die Software-Architektur SCA3D demonstrieren den Einsatz eines kombinierten Ansatzes für das verteilte Arbeiten mit 3D Dokumenten. Die 3D Dokumente werden dabei zwischen Server- und Client-Anwendung verteilt und sowohl entfernt als auch lokal visualisiert. Ein zentraler Aspekt ist hierbei die Verteilung und Unterstützung von beliebigen Objektrepräsentationen in der verteilten Umgebung, wobei die Verbindung zum Originaldokument solange wie nötig aufrechterhalten bleibt. Ein Ziel für zukünftige Arbeiten sollte daher die weitere Integration eines möglichst breiten Angebots von Objektrepräsentationen sein. Der Einsatz von und der Übergang zwischen parametrischen Objekten verschiedener Art, 3D Netzen mit beliebiger Auflösung, kombinierten Repräsentationsformen mittels Bounding-Box- und Bilddaten und schließlich 2D Bild- und Konturinformation sollte möglichst kontinuierlich im Arbeitsprozeß stattfinden. Ein wichtiger Punkt, der dafür weiter untersucht werden müßte, ist die Änderung der verschiedenen Objektrepräsentationen bei der Anwendung von geometrischen Manipulationen, so daß ein Originalobjekt und seine Repräsentationen in der Client-Szene möglichst lange übereinstimmen. Der Umweg über eine vollständige Aktualisierung des Client-Objektes durch Erzeugung einer neuen Objektrepräsentation am Server sollte dabei nur in Ausnahmefällen nötig werden, um die Belastung von Server-Rechner, Netz und Client-Rechner gering zu halten. Die Aufspaltung der Verzögerungszeit in ihre Komponenten, die durch Bildgenerierung am Server, Datenübertragung

und Bildanzeige am Client bestimmt werden, und ihre getrennte Berücksichtigung wäre hier sinnvoll, um eine individuelle Anpassung an die beteiligten Parameter wie Netzlast, Server-Last und Client-Last zu erlauben.

Dies führt direkt zu der Forderung nach effizienteren Codierungsverfahren für Bild- und Geometriedaten, die während des Arbeitsprozesses übertragen werden müssen. Im Rahmen diese Arbeit wurden sowohl Bildcodierverfahren der ersten Generation (MPEG-1, H.263) als auch solche der zweiten Generation (MPEG-4) getestet und in den verschiedenen Prototypen angewendet. Der aktuelle Stand spezieller Verfahren zur Kompression von Geometriedaten wurden nur am Rande verfolgt. Zur effizienten Übertragung der beschriebenen Objektrepräsentationen sollte der Einsatz dieser Verfahren weiter untersucht werden. Insbesondere Kompressionsverfahren, die Objektinformationen bei der Codierung erhalten und zur Steigerung des Kompressionsgrades verwenden, sind in diesem Zusammenhang sehr interessant.

Auch die weitere Ausarbeitung von dynamischen Schnittstellen zu beliebigen 3D Dokumenten ist vielversprechend. Hierbei ist die automatische Erzeugung von 3D Tools in der Client-Szene zur Manipulation von beliebigen Parametern der Dokumentobjekte für die praktische Anwendung und eine gute Benutzerführung besonders interessant. Wenn einem Benutzer bei der Bearbeitung eines ihm unbekannten 3D Dokumentes die passenden Manipulationswerkzeuge im Augenblick der Interaktion zur Verfügung gestellt werden, wäre eine Erkundung des Dokumentes und das Verstehen des Inhaltes sicher in verbesserter Weise möglich. Auch die weitere Differenzierung des Aufbaus der Client-Szene durch den Einsatz mehrerer Projektionsflächen, die in verschiedener Weise kombiniert werden können, würde die praktische Einsetzbarkeit des Systems erweitern.

Schließlich sollte mit Sicht auf die schnelle Entwicklung der Technologie im Bereich der vernetzten Informationssysteme im Internet angestrebt werden, eine möglichst weitreichende Verbreitung und Einsetzbarkeit des hier beschriebenen Systems zu erreichen, indem internationale Standards bei der Umsetzung verwendet werden. Zur allgemeinen Beschreibung von Dokumenten, um inhalts-basierte Merkmale wiederzugeben oder um auf entfernte Objekte zuzugreifen, wird in den letzten Jahren an einer Vielzahl von Standards gearbeitet. Die wichtigsten sind in diesem Zusammenhang der W3C Standard *XML* [GP01] zusammen mit dem objekt-orientierten Dokumentmodell *DOM* [DOM00], der ISO Standard *MPEG-7* [ISO00a] zur inhalts-basierten Beschreibung von beliebigen Dokumenten und Standards zur Realisierung von Remote-Procedure-Calls über WWW-eigene Protokolle, wie es etwa die Spezifikation *XML-RPC* [SJD01] oder der W3C Standard *SOAP* (*Simple Object Access Protocol*) [See01] ermöglichen. Die Erweiterung des Konzeptes auf allgemeine Dokumente, die neben 3D Objekten auch Audio-, Video-, Bild- und Textobjekte enthalten, ist durch Verwendung dieser Technologien für zukünftige verteilte Informationssysteme vielversprechend. Untersuchungen zur Erweiterbarkeit des Konzeptes auf allgemeine digitale Dokumente wären daher für zukünftige Arbeiten ein guter Ansatzpunkt mit weitreichenden Möglichkeiten.

Literaturverzeichnis

[Alh98] Alhir, S. *UML in a Nutshell,* O'Reilly, ISBN 1-56592-448-7, 1998

[AMD94] Ang, C.S., Martin, D.C., Doyle, M.D. *Integrated Control of Distributed Volume Visualization through the World Wide Web,* Proceedings of IEEE Visualization 94, 1994

[Atk99] Atkinson, L. *Core PHP programming: Using PHP to Build Dynamic Web Sites,* Markt und Technik, ISBN: 013020787X, 1999

[BB00] Benedens, O., Busch, C. *Towards Blind Detection of Robust Watermarks in Polygonal Models* Computer Graphics Forum, 19(3) (Proceedings of EUROGRAPHICS'00), 2000

[BBH00] Borowski, M., Bröcker, L., Heisterkamp, S., Karajannis, A., Kolb, I., Löffler, J. *ProDB-Projektdokumente intelligent verwalten,* Proceedings DOAG'2000, Deutsche Oracle Anwender Konferenz, pp. 847-857, 2000.

[BBL00] Borowski, M., Bröcker, L., Heisterkamp, S., Löffler, J. *Structuring the Visual Content of Digital Libraries using CBIR Systems,* IEEE International Conference on Information Visualization, 2000

[BDG98] Brodlie, K.W., Duce, D.A., Gallop, J.R., Wood, J.D. *Distributed cooperative visualization,* STAR: State of the Art Reports - Eurographics98. Eurographics Association, pp. 27-50, 1998

[Ben99] Benedens, O. *Watermarking of 3D-polygon-based models with robustness against mesh simplification,* Proceedings of SPIE Conference on Electronic Imaging '99, Security and Watermarking of Multimedia Contents, 1999

[Bis00] Bissel, T., Bogen, M., Hadamschek, V., Riemann, C. *Protecting a Museum's Digital Stock through Watermarks,* Proceedings of the International Conference Museums and the Web 2000, ISBN: I-885626-20-7, pp. 73-82, 2000

[BK96] Broll, W., Koop, T.: *VRML: Today and Tomorrow,* Computers & Graphics, Vol. 3(20). 1996, pp. 427-434, 1996

[BMN99] Booch, G. , Martin, R. , Newkirk, J. W. *Object-oriented Analysis and Design with Applications,* Addison Wesley Longman , ISBN 0-201-89551-X, 1999

[Cam98] Campagna, S. et al. *Enhancing Digital Documents by Including 3D-Models,* Computer & Graphics, Vol.22(6), pp. 655-666, 1998

[CC78] Catmull, E., Clark, J.H. *Recursively generated B-spline surfaces on arbitrary topological meshes,* Computer Aided Design, 10, pp. 350-355, 1978

[CCE98] Carraro, G., Cortes, M., Edmark, J., Ensor, J. *The Peloton Bicycling Simulator*, 3rd Symposium on the Virtual Reality Modeling Language VRML98, ACM SIGGRAPH, 1998

[CDK94] Coulouris, G., Dollimore, J., Kindberg, T. *Distributed systems concepts and design*, Addison-Wesley (International computer science series), ISBN 0-201-62433-8, 1994

[CH93] Carlsson, C., Hagsand, O. *DIVE- A Platform for Multi-User Virtual Environments*, Computer & Graphics, Vol.7(6), pp. 663-669, 1993

[CHRS98] Christiansen, A., Höding, M., Rautenstrauch, C., Saake, G. *Oracle8 - effizient einsetzen*, Addison-Wesley, ISBN 03-8273-1347-3, 1998

[CWH00] Campione, M. , Walrath, K. , Huml, A. *The Java tutorial: a short course on the basics*, Addison-Wesley (The Java series), ISBN 0-201-70393-9, 2000

[Del99] Del Bimbo, A. *Visual Information Retrieval*, Morgan Kaufmann Publishers, ISBN 1-55860-624-6, 1999

[Die01] Diehl, S. *Distributed Virtual Worlds*, Springer-Verlag, ISBN 3-540-67624-4, 2001

[Dit00] Dittmann, J. *Digitale Wasserzeichen*, Springer Verlag, ISBN 3-540-66661-3, 2000

[DOM00] DOM (W3C) *Document Object Model (DOM) Level 2 Core Specification*, W3C (World Wide Web Consortium) Document Object Model Level 2. Available at http://www.w3.org/TR/DOM-Level-2-Core

[DSS99] Dittmann, J., Steinmetz, A., Steinmetz, R. *Content-based Digital Signature for Motion Pictures Authentication and Content-Fragile Watermarking*, IEEE Multimedia Systems '99, Int. Conference on Multimedia Computing and Systems, Vol. 1, pp. 209 - 213, ISBN 0-7695-0253-9, 1999.

[EF00] Enders, A., Fellner, D.W. *Digitale Bibliotheken: Informatik-Lösungen für globale Wissensmärkte*, dpunkt-Verlag, ISBN 3-932588-77-0, 2000

[EHTE00] Engel, K., Hastreiter, P., Tomandl, B., Eberhardt, K. and Ertl, T. *Combining Local and Remote Visualization Techniques for Interactive Volume Rendering in Medical Applications*, In Procceedings of IEEE Visualization '00 , pp.6-10, 2000

[EK98] Ebrahimi, T., Kunt, M. *Visual data compression for multimedia applications*, Proceedings of the IEEE, Vol.86, No.6, 1109-1125, 1998

[ES98] Effelsberg, W., Steinmetz, R. *Video Compression Techniques*, dpunkt-Verlag, ISBN 3-920993-13-6, 1998

[ESE99] Engel, K., Sommer, O., Ernst, C. and Ertl., T. *Remote 3D Visualization using Image-Streaming Techniques*, In Advances in Intelligent Computing and Multimedia Systems (ISIMADE '99) , pp. 91-96, 1999

[FDFH90] Foley, J., van Dam, A., Feiner, S., Hughes, J. *Computer Graphics: Principles and Practice, Second Edition*, Addison-Wesley, Reading, Massachusetts, ISBN 0-201-84840-6, 1990

[FH97] Fellner, D.W., Hopp, A. *MRT-VR Multi-User Virtual Environment*, Proceedings of AAA 97, 1997

[FHM98] Fellner, D.W., Havemann, S., Müller, G. *Modeling of and Navigation in complex 3D Documents*, Computer & Graphics, Vol.22(6), pp. 647-653, 1998

[FKK96] Freier, A., Karlton, P. and Kocher, P. *The SSL 3.0 Protocol*, Netscape Communications Corporation, 1996.

[FZ99] Fellner, D.W., Zens, M. *Management and Workflow of Electronic Documents using a 2nd-Generation WWW-Server*, Proceedings WebNet 99, pp. 354-359, 1999

[GFG98] Goebbels, G., Fournier, N., Göbel, M. et al. *Remote Visualization of Radiological Data on a Responsive Workbench*, In Proceedings of CARS '99, Computer Assisted Radiology and Surgery, 1999

[Gla84] Glassner, A. *Space subdivision for for fast ray tracing*, IEEE Computer Graphics and Apllications, Vol.4 (10), pp. 15-22, 1994

[Gla99] Glassner, A. *Principles of Digital Image Synthesis*, Vol.2: 987-1052, Morgan Kaufmann Publishers, ISBN 1-55860-276-3, 1999

[GLG00] Goebbels, G., Lalioti, V., Göbel, M. *On Collaboration in Distributed Virtual Environments*, In The Journal of Three Dimensional Images, Japan, Vol.14(4), 12, 2000

[GP01] Goldfarb, C., Prescod, P. *The XML handbook*, Prentice-Hall, ISBN 0-13-055068-X, 2001

[Gro98] Gross, M.H., Kobbelt, L. et al. *Hierarchical Methods for Computer Graphics*, Eurographics '98, Lisbon, EG Tutorial Notes, 1998

[GTGB84] Goral, C., Torrance, K., Greenberg, D. and Battaile, B. *Modelling the interaction of light between diffuse surfaces*, Computer Graphics (Proceedings Siggraph '84), Vol.18(3), pp. 213-222, 1984

[HCF97] Hamilton, G. , Cattell, R. , Fisher, M. *JDBC database access with Java: a tutorial and annotated reference* Addison-Wesley (The Java series), ISBN 0-201-30995-5, 1997

[HEG98] Hartung, F., Eisert, P., Girod, B. *Digital Watermarking of MPEG-4 Facial Animation Parameters*, In Computer & Graphics, Vol. 22(4), pp. 425-435, 1998

[HG00] Hubeli, A., Gross, M. *A survey of Surface Representations for Geometric Modeling*, Technical Report, ETH Zürich, 2000

[HH95] Hartung, F., Horn, U. *Codierung digitaler Videosignale nach dem MPEG- Standard*, Digitaltechnik, Bd. 9, Heft 1, 18-23, 1995.

[HJT98] Haulsen, I., Jung, T., Tuchtenhagen, D. *Remote Control of Virtual Environments Using Image Streams*, Proceedings of Second IMA Conference on Image Processing, Leicester, United Kingdom, 1998

[HM90] Haber, R.B. and McNabb, D.A. *Visualization Idioms: A Conceptual Model for Scientific Visualization Systems*, Visualization in Scientific Computing , IEEE Computer Society Press, pp. 74-93, 1990

[Hop96] Hoppe, H. *Progressive meshes*, Computer Graphics (SIGGRAPH 1996 Proceedings), pages 99-108, 1996

[HS98] Hesina, G., Schmalstieg, D. *A Network Architecture for Remote Rendering*, Proceedings of 2nd International Workshop on Distributed Interactive Simulation and Real Time Applications (DIS-RT'98), pp. 88-91, 1998.

[HS00] Heuer, A., Saake, G. *Datenbanken: Konzepte und Sprachen*, International Thomson Press, ISBN 3-8266-0619-1, 2000

[HSF99] Hesina, G., Schmalstieg, D., Fuhrmann, A., Purgathofer, W. *Distributed Open Inventor: A Practical Approach to Distributed 3D Graphics*, Proceedings of ACM Virtual Reality Software & Technology '99 (VRST'99), pp. 74-81, London, 1999.

[ISO99a] ISO/IEC *MPEG-4: Coding of audio-visual Objects*, ISO/IEC International Standards 14496, Parts 1-6, 1999

[ISO00a] ISO/IEC *Overview of the MPEG-7 Standard*, ISO/IEC JTC1/SC29/WG11 N3445, Geneva, May/June, 2000

[Jac92] Jacobson, I. et al. *Object-oriented Software Engineering: A Use Case Driven Approach*, ISBN 0-201-54435-0, Addison-Wesley, 1992

[KP00] Katzenbeisser, S., Petitcolas, F. *Information Hiding techniques for stenography and digital watermarking*, Artech House computing library, ISBN 1-58053-035-4, 2000

[Kum96] Kumar, V. *MBone: Interactive Multimedia on the Internet*, New Riders Publishing, ISBN 1-56205-397-3, 1996

[LBS98] Lewis, G., Barber, S., Siegel, E. *Programming with Java IDL*, Wiley , ISBN 0-471-24797-9, 1998

[LeG92] Le Gall, D.J. *The MPEG video compression algorithm*, Signal Processing: Image Communication, Vol.4,No.4, 129-140, 1992

[Lev00] Levoy, M. et al. *The Digital Michelangelo Project: 3D Scanning of Large Statues* Proceedings of SIGGRAPH'2000, 2000

[LF01] Löffler, J. and Fellner, D.W. *Adaptive Visualization of distributed 3D Documents using Image Streaming Techniques*, Springer (Wien), Proceedings of Eurographics Multimedia Workshop 2001 (Manchester UK), ISBN 3-211-83769-8, 2001

[LFe01] Löffler, J. and Fellner, D.W. *A Software Architecture for Adaptive Visualization of Distributed 3D Documents in Open Information Spaces*, Proceedings of Cast'01, Conference on artistic, cultural and scientific aspects of experimental media spaces, Germany, 2001

[LHG97] Larkin, S., Hewitt, W.T., Grant, A.J. *Systems and Architectures for Visualization*, Proceedings from the 1997 CERN School of Computing, Prague, Czech Republic, 1997.

[Lia99] Liang, S. *Java Native Interface: programmer's guide and specification*, Addison Wesley Longman (Java Series), ISBN 0-201-32577-2, 1999

[Loe00a] Löffler, J. *Object-based Image Coding for Cooperative 3D Visualization*, Proceedings of International Conference on Computer Graphics, Visualization and Interactive Digital Media (WSCG 2000), ISBN 80-7082-612-6, pp. 197-203, 2000.

[Loe00b] Löffler, J. *Content-based Retrieval of 3D Models in Distributed Web Databases by Visual Shape Information,* Proceedings of IEEE International Conference on Information Visualization (IV 2000), pp. 82-87, London, 2000.

[Loop87] Loop, C.T. *Smooth Subdivision Surfaces Based on Triangles,* PhD Thesis, Department of Mathematics, University of Utah, 1987

[Lyn97] Lynch, C. *The Z39.50 Information Retrieval Standard. Part 1, A Strategic View of Its Past, Present and Future,* In D-Lib Magazine (online), April, 1997

[MB94] Macedonia, M., Brutzmann, D. *MBONE, the Multicast Backbone,* IEEE Computer, Vol. 27(4), pp. 30-36, 1994

[MCB98] Miller, M., Cox, I., Bloom, J. *Watermarking in the real world: an application to DVD,* Workshop Security Issues in Multimedia Systems, Proceedings of IEEE Multimedia Systems Conference '98, 1998

[MCO97] Mann, Y., Cohen-Or, D. *Selective Pixel Transmission for Navigating in Remote Virtual Environments,* Proceedings of EUROGRAPHICS'1997 Conference, Vol.16, No. 3, 1997

[MF99] Müller, G., Fellner, D.W. *Hybrid Scene Structuring with Application to Ray Tracing,* Proceedings of International Conference on Visual Computing (ICVC'99), pp. 19-26, Goa, India, 1999

[MH00] Müller, K., Havemann, S.. *Subdivision Surfaces Tessellation on the Fly using a versatile Mesh Data Structure,* Proceedings of EUROGRAPHICS'2000 Conference, Vol.19, No. 3, 2000

[NN85] Nishita, T. and Nakamae, E. *Continuous tone representation of three-dimensional objects taking account of shadows and interreflection,* Computer Graphics (Proceedings Siggraph '85), Vol.19(3), pp. 23-30, 1985

[NQ98] Nahrstedt, K., Qiao, L. *Multimedia-secure Gateway,* Workshop Security Issues in Multimedia Systems, in Proceedings od IEEE Multimedia Systems Conference IMCMS'98, 1998

[OMA98] Ohbuchi, R., Masuda, M., Aono, M. *Watermarking Three-dimensional Polygonal Models through Geometric and Topological Modifications,* Proceedings of IEEE Journal on Selected Areas in Communications, pp. 551-560, 1998

[OMG98] Object Management Group (OMG) *The Common Object Request Broker: Architecture and Specification,* Revision 2.3, 1998

[Ora99] *Oracle8i Visual Information Retrieval User's Guide and Reference,* Oracle 8.1.5 Documentation, 1999

[Pho75] Phong, B.-T. *Illumination for computer generated pictures,* Communications of the ACM, Vol.18(6), pp. 311-317, 1975

[PR99] Paquet, E., Rioux, M. *The MPEG-7 Standard and the Content-based-Management of Three-dimensional Data: A Case Study,* International Conference on Multimedia Computing and Systems: 375-380, 1999

[PR01] Puder, A., Römer, K. *Middleware für verteilte Systeme*, dpunkt-Verlag, ISBN 3-932588-03-7, 2001

[RFC1521] Borenstein N. and Freed, N. *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, RFC 1521, 1993.

[RFC2616] Fielding R. et al. *Hypertext Transfer Protocol - HTTP/1.1*, RFC2616, 1999

[RFL98] Rantzau, D., Frank, K., Lang, U., Rainer, D., Wössner, U. *COVISE in the CUBE: An Environment for Analyzing Large and Complex Simulation Data*, Proceedings of the 2nd Workshop on Immersive Projection Technology (IPT '98), 1998

[RMW98] Rzepa, H. S., Murray-Rust, P. and Whitaker, B. J. *The Application of Chemical Multipurpose Internet Mail Extensions (Chemical MIME) Internet Standards to Electronic Mail and World-Wide Web information exchange*, Journal of Chemical Information and Computer Sciences, Vol.38, 976-982, 1998

[RR97] Riley, M., Richardson, I. *Digital video communications* Artech House , ISBN 0-89006-890-9, 1997

[Sch00] Schäfer, S. *Efficient Object-Based Hierarchical Radiosity Methods*, Dissertation am Fachbereich Mathematik und Informatik der Technischen Universität Braunscheig, 2000

[SG96] Schmalstieg, D., Gervautz, M. *Demand-Driven Geometry Transmission for Distributed Virtual Environments* Computer Graphics Forum, 15, 3 (Proceedings of EUROGRAPHICS'96, Poitier, France), pp. 421-433, 1996

[See01] Seely, S. *SOAP: cross plattform*, Prentice-Hall, ISBN 0-13-090763-4, 2001

[Sieg96] Siegel, J. *CORBA: fundamentals and programming*, New York, N.Y. (u.a.) : Wiley , 1996. - XXVI, ISBN 0-471-12148-7, 1996

[Sik97] Sikora, T. *The MPEG-4 Video Standard Verification Model*, IEEE Transactions on circuits and systems for video technology, Vol.7(1), 19-31, 1997

[SJD01] Saint Laurent, S., Johnston, J., Dumbill, E. *Programming web services with XML-RPC*, O'Reilly Verlag, ISBN 0-596-00119-3, 2001

[SRD98] Sowizral, H. , Rushforth, K. , Deering, M. *The Java 3D API specification*, Addison-Wesley , ISBN 0-201-32576-4, 1998

[SSNM97] Schmidt, J.W., Schröder, G. et al. *Linguistic and architectural requirements for personalized digital libraries*, Int. J. of Digital Libraries 1, 1, 89-104, 1997

[Stei00] Steinmetz, R. *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*, Springer-Verlag, ISBN 3-540-67332-6, 2000

[SZ99] Singhal, S., Zyda, M. *Networked Virtual Environments: Design and Implementation*, Addison-Wesley, ISBN 0-201-32557-8, 1999

[VY99] Vaysburd, A. and Yajnik, S. *Exactly-Once End-to-End Semantics in CORBA Invocations Across Heterogeneous Fault-Tolerant ORBs*, Proceedings of 18th IEEE Symposium On Reliable Distributed Systems, pp. 296-297, 1999

[Wah98] Wahl, G. *UML kompakt*, Zeitschrift OBJEKTspektrum 2/98, 101Communications Verlag, 1998

[War94] Ward, G.J. *The RADIANCE lighting simulation and rendering system*, Computer Graphics, Proceedings of SIGGRAPH‘94, 459-472, 1994

[Wer98] Wernecke, J. *The Inventor Mentor - Programming Object-Oriented 3D Graphics with Open Inventor*, Addisson Wesley, ISBN 0-201-62495-8, 1998

[Whi98] Whitted, T. *An improved illumination model for shaded display,* Communications of the ACM, Vol.23, No.6, pp. 343-349, 1980

[WKL98] Weibel, S., Kunze, J., Lagoze, C. et al. *Dublin Core metadata for Resource Discovery,* IETF Informational RFC, (http://www.ietf.org/rfc/rfc2413.txt), 1998

[WK96] White, J.S., and Klaphaak D.T. *Using Java and Open Inventor to Rapid Prototype 3D Worlds*, Proceedings of 3D and Multimedia on the Internet, WWW and Networks, Bradford, UK, 1996

[WWB96] Wood, J., Wright, H., Brodlie, K. *Visualization over the WWW and its application to environmental data,* Proceedings of IEEE Visualization 96, pp. 81-92, 1996.

[YC98] Yi, J., Chelberg, D. *Model-based 3D Object Recognition Using Bayesian Indexing,* Computer Vision and Image Understanding, Vol.69, No.1: 87-105, 1998